Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Trabalho Prático

Jogo de Xadrez

Unidade Curricular: Programação Avançada

ANO LETIVO 2024/2025

Beatriz Filipa Santos Marques - 2022137934

Alexandre Fraga Gomes - 2018015925

Pedro Teixeira Amorim - 2022157609

Índice

1. Introdução

O seguinte relatório surge no âmbito do projeto final da unidade curricular de programação avançada e tem por objetivo descrever o desenvolvimento de um jogo de xadrez em Java, utilizando os princípios da programação orientada a objetos, a biblioteca JavaFX, e vários padrões de design abordados ao longo do semestre.

A aplicação desenvolvida permite realizar um jogo xadrez entre dois jogadores, suportando todas as regras fundamentais do jogo, incluindo movimentos específicos de cada peça, captura, promoção de peões, verificação de checkmate e empate (stalemate). Adicionalmente, foram implementadas funcionalidades extra como a gravação e carregamento de jogos, importação e exportação parciais, um modo de aprendizagem com suporte a undo/redo, e a sinalização visual das jogadas possíveis, de acordo com o solicitado ao longo das metas semanais.

A estrutura do projeto foi organizada segundo o padrão arquitetural MVVM, garantindo uma separação clara entre a lógica - *modelo* - a interface gráfica - *ui* - e a camada de mediação e comunicação entre ambos - *facade.* O projeto foi desenvolvido de forma coerente, acompanhando as etapas propostas ao longo do semestre.

Este relatório detalha as principais decisões de implementação, os padrões utilizados, o funcionamento e relação entre as classes, e apresenta uma análise do cumprimento dos requisitos definidos no enunciado.

2. Implementação

O projeto foi desenvolvido na integra em Java, respeitando a estrutura modular definida no enunciado. A implementação seguiu uma abordagem orientada a objetos, com uma divisão clara entre o modelo de dados, a interface gráfica e a lógica de mediação.

A arquitetura geral baseia-se no padrão MVVM *(Model-View-ViewModel)*, utilizando o padrão *facade* para centralizar a lógica da aplicação e o padrão *observer (PropertyChangeSupport)* para manter a interface gráfica sincronizada com o estado do jogo. Foram ainda utilizados os padrões *memento*, para gestão de histórico das jogadas (undo/redo).

A implementação foi realizada de forma incremental, acompanhando as etapas propostas nas aulas práticas, com recurso ao GitHub para controlo de versões, colaboração e entrega.

2.1. Decisões de implementação

Durante o desenvolvimento do nosso trabalho, foram tomadas várias decisões importantes com impacto na organização e funcionamento da aplicação. Estas, contribuíram para um código modular, reutilizável, testável e facilmente extensível, respeitando as boas práticas consolidadas ao longo do semestre nas aulas.

2.1.1. Separação de responsabilidades

O projeto foi dividido em 3 sub-packages principais (model, ui, memento) de forma a garantir a independência entre a lógica de jogo e a interface gráfica, conforme recomendado.

Foi também, como referido anteriormente, utilizada uma classe *Facade* (ChessGameManager) onde toda a interação da interface com a lógica do jogo é feita exclusivamente através da mesma, que funciona como ponto central de acesso às operações disponíveis, simplificando a comunicação com o modelo.

2.1.2. Notificações assíncronas

A atualização da interface gráfica é feita com base em eventos de propriedade (PropertyChangeSupport), permitindo uma desagregação entre a UI e a lógica, e atualização automática de elementos gráficos como o tabuleiro, nomes dos jogadores e jogadas válidas.

2.1.3. Modo de aprendizagem

Como solicitado nas etapas semanais, foi implementado um modo de aprendizagem especial que permite o uso das funcionalidades de undo e redo, utilizando o padrão *memento* para guardar/restaurar estados anteriores do jogo.

2.1.4. Canvas

Foi utilizado o componente Canvas da biblioteca JavaFX como base para a construção do tabuleiro do jogo de xadrez. A escolha do Canvas permite um controlo direto sobre a renderização gráfica, tornando-o ideal para desenhar dinamicamente os elementos visuais do xadrez, como o fundo do tabuleiro, as casas claras e escuras, e as peças de cada jogador.

2.1.5. Persistência de dados

O nosso trabalho permite guardar e abrir jogos completos usando serialização (ChessGameSerialization), bem como importar e exportar jogos parciais em ficheiros CSV/texto, reforçando a flexibilidade da aplicação, tal como pedido.

3. Diagramas de Padrões

Neste segmento do relatório, apresentam-se os principais padrões de design utilizados no desenvolvimento da aplicação de xadrez. Os diagramas abaixo ilustram, de forma clara e concisa, como esses padrões foram aplicados na arquitetura do sistema.

- **Factory**: cria dinamicamente peças de xadrez com base no tipo ou formato;
- **Memento**: permite desfazer/refazer jogadas sem expor o estado interno do jogo;
- **Observer**: notifica automaticamente a interface sempre que há mudanças no estado do jogo ou nos logs;
- **Singleton**: garante que o gestor de logs (ModelLog) tenha apenas uma instância partilhada;
- **Facade**: simplifica a comunicação entre a interface gráfica e a lógica de jogo;

3.1. Factory

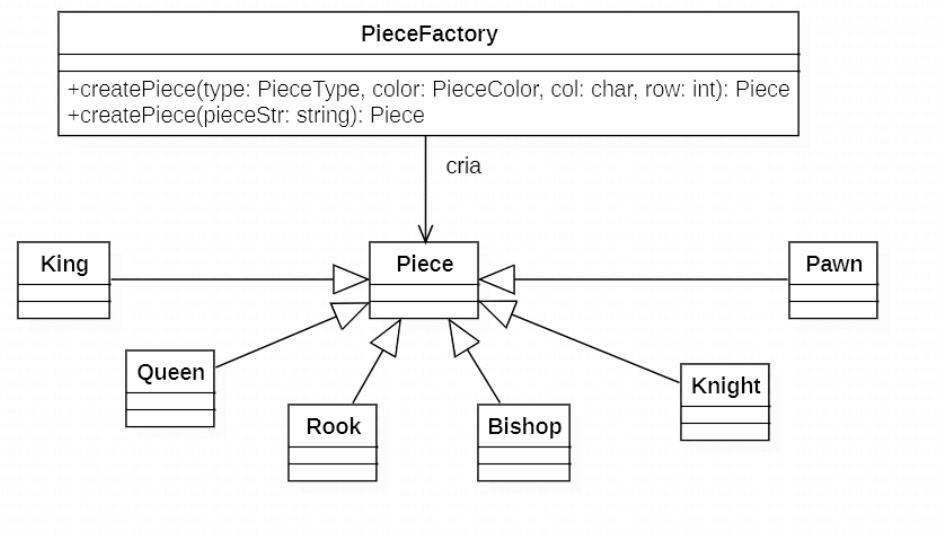


Figura 1 : Padrão Factory

3.2. Memento

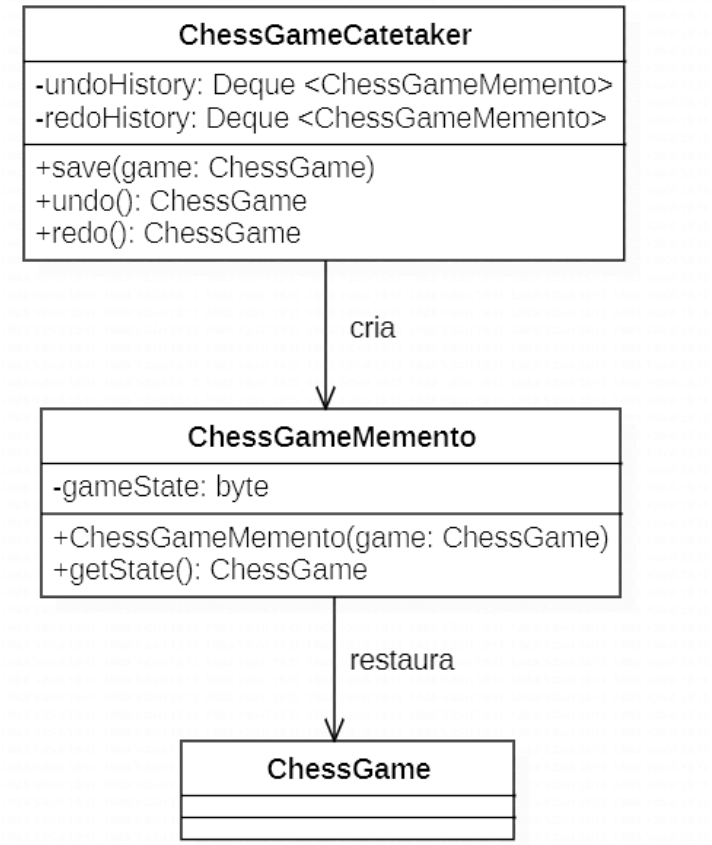


Figura 2 : Padrão Memento

3.3. Observer

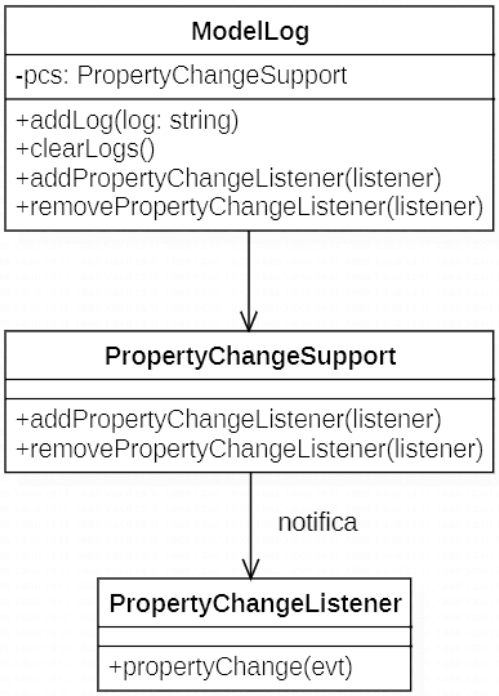


Figura 3 : Padrão Observer

3.4. Singleton

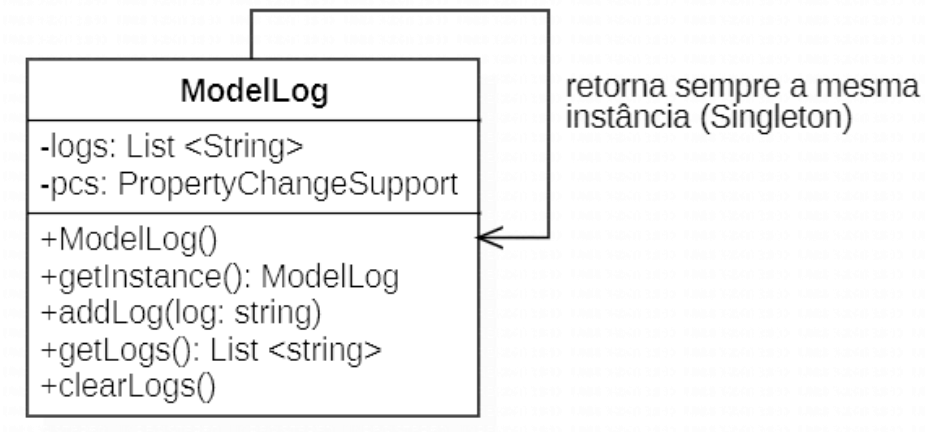


Figura 4 : Padrão Singleton

3.5. Facade

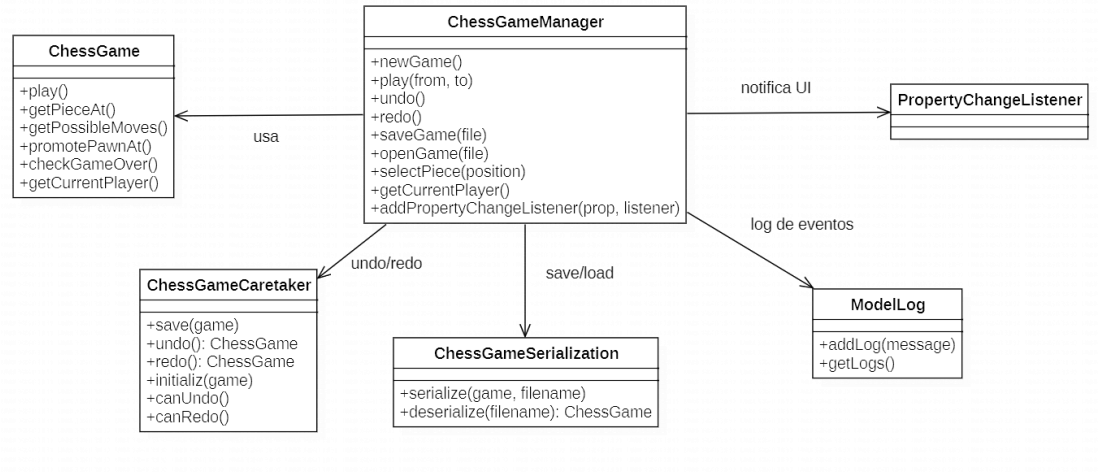
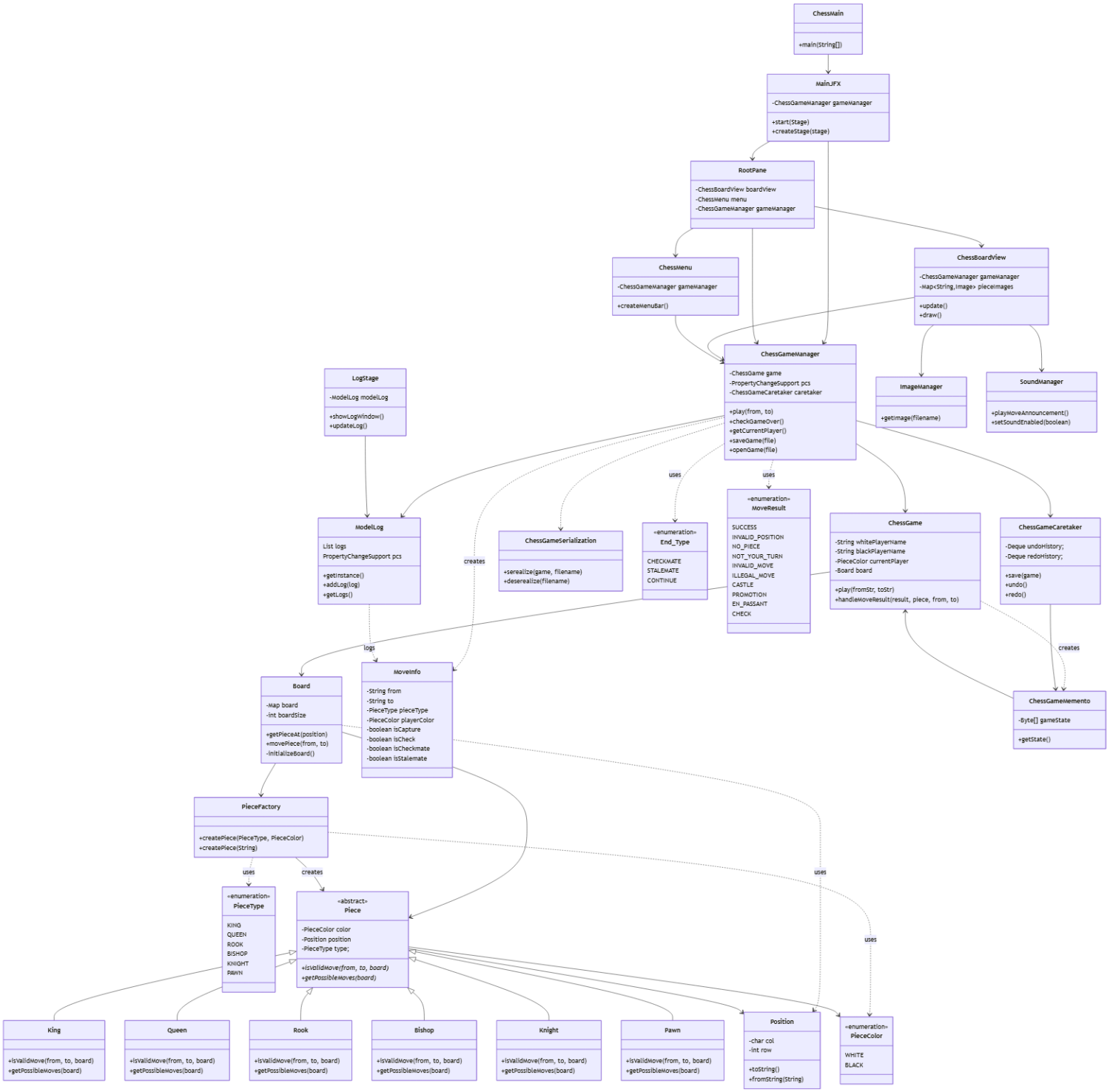


Figura 5 : Padrão Facade

4. Relação entre classes



5. Descrição de classes

Tal como solicitado, a tabela seguinte apresenta uma descrição sucinta das principais classes utilizadas no desenvolvimento do jogo de xadrez. Cada classe foi projetada com uma responsabilidade específica, contribuindo para a organização, modularidade e extensibilidade do sistema.

Nesta tabela, indica-se o papel de cada classe no domínio do jogo, bem como o seu objetivo funcional dentro da arquitetura geral da aplicação. Esta visão ajuda a compreender como os componentes se relacionam e colaboram na execução da lógica do jogo, gestão de estado, persistência e interação com a interface gráfica.

5.1. Tabela de classes

CLASSE	DESCRIÇÃO	OBJETIVO/FUNÇÃO PRINCIPAL
CHESSGAMEMANAGER	Gestor central do jogo	Atua como Facade e ViewModel, gerindo a lógica, estado e notificações
CHESSGAME	Representa o estado e regras do jogo de xadrez	Contém toda a lógica das jogadas, regras, fim de jogo, promoção, etc.
CHESSGAMECARETAKER	Guarda estados do jogo	Implementa o padrão Memento para suporte a undo/redo
CHESSGAMEMEMENTO	Captura e guarda o estado serializado do jogo	Permite restaurar estados anteriores do jogo
CHESSGAMESERIALIZATION	Class de serialização	Guarda e carrega o estado do jogo para ficheiros
MODELLOG	Gestor de logs de eventos do modelo	Classe Singleton com suporte a Observer para a interface
PIECEFACTORY	Fábrica de peças de xadrez	Cria instâncias de Piece dinamicamente com base em tipo ou string
PIECE (E SUBCLASSES)	Representa as peças de xadrez (King, Queen, etc.)	Modela o comportamento individual de cada tipo de peça
BOARD	Representa o tabuleiro do jogo	Armazena as peças e as suas posições; valida ataques, movimentos, etc.
POSITION	Representa uma posição no tabuleiro	Abstrai a conversão entre coordenadas ("e2", col, row)
CHESSBOARDVIEW	Componente gráfico personalizado do tabuleiro	Desenha o tabuleiro, peças, movimentos e interações visuais com o jogador
ROOTPANE	Estrutura principal da interface gráfica	Organiza visualmente o tabuleiro, toolbar, menus e labels dos jogadores
CHESSMENU	Barra de menus da interface	Oferece ao utilizador opções de jogo, modos, atalhos e comandos como Undo

6. Funcionalidades Implementadas

FUNCIONALIDADE	ESTADO	OBSERVAÇÕES
CRIAR NOVO JOGO COM NOMES DE JOGADORES	Implementado	Os nomes são pedidos através do TextInputDialog no início de cada jogo.
JOGADAS VÁLIDAS SEGUNDO REGRAS DO XADREZ	Implementado	Toda a lógica de movimentos está nas classes ChessGame e Piece.
XEQUE, XEQUE-MATE E STALEMATE	Implementado	Detetado automaticamente após cada jogada; mensagens de fim aparecem.
PROMOÇÃO DE PEÃO	Implementado	Interface mostra diálogo com imagens para promoção ao chegar à última linha.
IMPORTAR/EXPORTAR JOGO PARCIAL (TEXTO/CSV)	Implementado	Utiliza .txt com representação CSV.
GUARDAR/ABRIR JOGO COMPLETO (FICHEIRO SERIALIZADO)	Implementado	Classe ChessGameSerialization.
UNDO/REDO	Implementado	Ativo apenas no modo de aprendizagem, com histórico gerido pelo memento.
MODO APRENDIZAGEM	Implementado	Ativa funcionalidades como Undo/Redo e possíveis jogadas.
MOSTRAR POSSÍVEIS JOGADAS	Implementado	Apenas visível no modo aprendizagem, via CheckMenuItem.
SONS DE JOGADA	Implementado	Sons diferentes consoante o tipo de jogada (xeque, captura, etc.).
DETETAR JOGADAS INVÁLIDAS	Implementado	Jogadas ilegais ou fora da vez são ignoradas; posição fica realçada a vermelho.
INTERFACE GRÁFICA INTERATIVA EM JAVAFX	Implementado	Inclui menus (ChessMenu), labels, atalhos e desenho do tabuleiro.
LOG DE EVENTOS DO JOGO	Implementado	Através da classe singleton ModelLog.
ARQUITETURA MVVM	Implementado	Separação clara entre interface (RootPane/ChessBoardView) e lógica (ChessGameManager).

7. Conclusão e considerações finais

O desenvolvimento deste jogo de xadrez permitiu consolidar conhecimentos fundamentais sobre java, interfaces gráficas com JavaFX e, especialmente, a aplicação de padrões de design de software em projetos reais.

Durante a implementação, foi possível:

- Criar uma estrutura modular e extensível com base na arquitetura MVVM (Model-View-ViewModel);
- Aplicar com sucesso padrões como Facade, Memento, Factory, Observer e Singleton, que contribuíram significativamente para a organização e manutenção do código;
- Desenvolver uma interface gráfica intuitiva, com suporte a funcionalidades como jogadas válidas, promoção de peões, undo/redo, modo aprendizagem, sugestão de jogadas e sons de jogada, tornando a experiência do utilizador mais completa.

Além disso, foram utilizados conceitos avançados como:

- Serialização de objetos para guardar e recuperar o estado completo do jogo;
-
- Notificação reativa via PropertyChangeSupport, assegurando que a interface se atualiza automaticamente ao longo do jogo.

Em suma, o trabalho resultou numa aplicação funcional, estável e com uma base sólida de design, refletindo boas práticas de engenharia de software. As funcionalidades foram implementadas com sucesso.