Trabalho de Programação Orientada a Objetos

Projeto de Xadrez

Ana Clara Santana Matta - 822396

Bianca Ribeiro Barros – 822806

Gabriela Aya Tiba – 820718

# Sobre o projeto e as dificuldades encontradas

Este relatório documenta o desenvolvimento de um jogo de xadrez em Java, utilizando princípios de Orientação a Objetos. O sistema modela elementos como jogadores, peças, tabuleiro e as suas casas, permitindo a execução de jogadas válidas, detecção de xeque e xeque-mate, além de funcionalidades para salvar e carregar jogos. O diagrama de classes anexo ilustra a estrutura do projeto, com destaque para as principais classes: **Gerenciador, Jogo, Jogador, Tabuleiro** e **Peças.**

Para o trabalho em grupo optamos por utilizar do controle de versionamento Git e a plataforma GitHub para armazenamento das versões.

Como o projeto foi desenvolvido por 3 pessoas, 2 ficaram responsáveis pela estruturação geral do código, como modelagem de classes e suas funcionalidades, e uma terceira pessoa ficou responsável por realizar o controle de versionamento, testagem de código e documentação.

Não adotamos uma divisão de tarefas no sentido onde cada membro fica responsável por uma classe ou método em específico, e sim optamos por conversar sobre o que cada membro se sentia capaz de desenvolver e fomos moldando nosso projeto em cima disso.

**Sobre as dificuldades encontradas:**

Por se tratar de um trabalho extenso e complexo, tivemos muitas dificuldades no que diz respeito ao início da estruturação das classes e seus métodos. Compreender, além dos métodos requisitados do projeto, quais outros métodos poderiam fazer parte de cada classe e como implementá-los.

Outra dificuldade recorrente foi a implementação da verificação do xeque e xeque-mate. Como realizar as simulações de jogadas para verificar cada caso onde o rei pode ou não escapar de um xeque.

# Diagrama Completo Finalizado

As linhas em azul representam novas relações criadas entre as Classes.

|  |
| --- |
| Novo diagrama de Classes completo - Pode ser visualizado melhor no documento “Diagrama Completo Xadrez.pdf” fornecido com os arquivos de projeto |

# Escopo do Software

* **Gerenciador:** responsável por iniciar o jogo. Exibe um menu inicial onde o usuário pode escolher entre iniciar uma nova partida ou continuar uma partida existente, lendo de um arquivo.

Possui métodos para salvar a partida atual ao parar o jogo, criando um arquivo caso não exista, ou salvando no arquivo já existente com o mesmo nome. Faz também o carregamento de partidas salvas da mesma maneira.

Possui, também, um método de teste para testar uma rodada de jogos onde são testados elementos como movimentação de peças, captura de peças, jogadas inválidas ou fora do tabuleiro, xeque e xeque-mate.

* **Jogo:** gerencia a lógica central do xadrez. Essa classe cria e estrutura o tabuleiro, colocando as peças em seus devidos lugares iniciais, além de estabelecer as cores dos jogadores e o estado do jogo. Esse estado sendo 1 para o início, 2 para identificação de xeque e 3 para a identificação de xeque-mate.

Além disso, é responsável por validar as jogadas, atualiza o estado da partida e detecta condições de xeque e xeque-mate.

* **Jogada:** armazena as informações de uma jogada específica, como a posição de origem e destino das peças, o jogador responsável pela jogada e o caminho percorrido pela peça. Além disso, contém métodos para validar jogadas e verificar condições especiais como xeque e xeque-mate.

Para verificar o xeque é necessário saber a posição do rei do adversário à partir da posição da sua posição no tabuleiro. É feita então uma jogada simulada para verificar se é valida. Se a jogada for válida temos um Xeque.

Para verificar o xeque-mate, pegamos a posição do rei do oponente e simulamos e temos todos os movimentos que ele poderia fazer para se livrar do xeque. Se todos esses movimentos forem inválidos, ou tirar o rei de Xeque temos um Xeque-Mate.

Por último, ela cria um caminho a partir da origem e destino da peça de uma jogada.

* **Jogador:** recebe e armazena informações dos jogadores da partida, como seu nome e sua cor no tabuleiro. Cada jogador é uma instância dessa classe. A classe também recebe e armazena as peças do jogador ao montar o tabuleiro, mantém informações sobre as peças ativas e capturadas pelo adversário. É responsável também por receber a jogada a ser realizada pelo jogador.
* **Tabuleiro:** modela o tabuleiro do jogo composto por uma matriz de Classes Casa. A classe controla a ocupação das casas, desenha o tabuleiro e valida se posições estão dentro dos limites permitidos. Também oferece métodos para localizar o rei de cada jogador, essencial para verificação do xeque.

O método para encontrar o rei no tabuleiro, usa a cor da peça como parâmetro. Percorre por toda a matriz de casas verificando se a peça na casa é uma instância da classe Rei e possui a cor do parâmetro. Se encontrar uma casa cuja peça ocupante correspondem a essas verificações, o método retorna a casa correspondente a esta peça.

Para desenhar o tabuleiro, e destacar as linhas e colunas utilizamos Strings de código de cores ANSI que imprimem as linhas e colunas em azul. Fazendo um desenho simulando um tabuleiro de xadrez com os desenhos recebidos de cada casa da classe Casa e de cada peça recebidos pela classe Peça.

* **Casa:** representa cada posição individual no tabuleiro. Guarda informações da cor da peça na casa, se a casa está ocupada, qual peça está ocupando a casa, a linha e a coluna em que a casa se encontra. Realiza as ações de ocupar e desocupar uma casa, além de verificar se a casa está dentro dos limites do tabuleiro.
* **Caminho:** responsável por guardar as casas que compõem o trajeto recebido de uma peça durante a jogada. Contém a sequência de casas entre a posição inicial e a final, verificando se estas estão livres ou ocupadas por outras peças.
* **Peça:** a classe peça é uma classe abstrata que guarda informações gerais de todas as peças como cor e se está em jogo ou foi capturada. Todas as peças derivadas desta base possuem métodos *sobrescritos* para verificar, conforme as regras de sua movimentação, se a movimentação é válida ou não. A classe peça também é responsável por criar, em String, o trajeto realizado pela peça da sua posição inicial para a sua posição final. Este método é utilizado pela classe Caminho para armazenar as casas que compõem este trajeto.

A Classe Derivada **Peão,** em específico, possui um método para checar se o ataque de um peão é válido ou não.

Para um peão branco, o ataque deve ser uma diagonal à frente (ou seja, a linha de destino deve estar uma casa acima da linha de origem e a coluna deve mudar para esquerda ou direita).

Para um peão preto, o ataque deve ser uma diagonal para trás (a linha de destino estar uma casa abaixo da linha de origem, e a coluna muda de forma semelhante).

# Abordagem de Testes

Durante o desenvolvimento do nosso projeto, realizamos uma série de testes para garantir a integridade e a funcionalidade adequada de todas as partes do sistema. A abordagem de testes incluiu:

* Testar o encapsulamento das classes Jogador, Casa, Caminho, Tabuleiro, Jogada, Jogo e das Peças para garantir que todos os métodos auxiliares fossem privados e todos os atributos fossem privados acessíveis somente via métodos públicos que realizassem seu tratamento.
* Realizamos testes para assegurar as interações entre as classes como, por exemplo, ao receber um trajeto de uma peça podemos formular uma lista de cassas que compõem um caminho percorrido por ela.
* Testamos diversos fluxos de jogos, checando o tratamento de erros para caso as casas fossem fora do limite do tabuleiro, se já estivessem ocupadas por outra peça. Dentro deste escopo também testamos a validação de movimento de peças para garantir que se movimentassem apenas no sentido permitido.
* Por fim, o teste mais realizado foi para garantir a lógica por trás do xeque e xeque-mate. Realizamos diversas partidas diferentes onde ocorriam o xeque e o xeque-mate.

O método *testeJogo()* presente na classe Gerenciador é uma simulação de partida com jogadas pré-definidas para a validação de todas as implementações realizadas.

Neste método, criamos uma instância do jogo com dois jogadores fictícios e executamos uma série de jogadas que cobrem diferentes aspectos do xadrez, incluindo movimentos normais, captura de peças, verificação de xeque, xeque-mate e verifica os casos de erro onde o jogador realiza uma jogada inválida (que não pode acontecer seja por conta da casa estar ocupada ou o movimento da peça) ou uma jogada fora dos limites do tabuleiro.

O método verifica a atualização de estados do jogo após cada jogada e confirma quando um jogo termina corretamente após alcançar um xeque-mate.

Após realizarmos esses testes, obtivemos os resultados esperados e concluímos que o projeto está pronto.

# O que não foi implementado

# Condições que podem causar erro

# Instruções de funcionamento