Trabalho de Programação Orientada a Objetos

Projeto de Xadrez

Ana Clara Santana Matta - 822396

Bianca Ribeiro Barros – 822806

Gabriela Aya Tiba – 820718

# Instruções de funcionamento

Antes de começar, certifique-se de que você tenha o JDK instalado para poder compilar e rodar o jogo em seu computador. Caso ainda não tenha, você pode baixá-lo [aqui](https://www.oracle.com/java/technologies/downloads/).

Passo a passo para compilar e executar o jogo:

Caso já possua uma IDE que compile arquivos Java:

1. Abra o diretório “xadrez” onde estão localizados os arquivos .java do projeto.
2. Abra o arquivo Gerenciador.java e clique em “compilar” ou “rodar” em sua IDE.

Caso não possua uma IDE, siga os passos abaixo:

1. Abra um terminal / console. Você pode encontrar digitando “terminal”, “cmd”, ou PowerShell em seu computador.
2. Navegue até o diretório onde se encontra a pasta “xadrez” onde os arquivos .java estão localizados.
   1. Você pode fazer isso navegando da pasta que vc se encontra até o diretório onde se encontra o diretório xadrez através do comando “cd nomediretorio” para entrar em um diretório, ou “cd ../” para sair de um diretório.
3. Execute o seguinte comando no terminal para compilar:

|  |
| --- |
| javac xadrez/\*.java |

Isso irá compilar todos os arquivos Java do projeto.

1. Após a compilação, execute o jogo com o seguinte comando:

|  |
| --- |
| java xadrez.Gerenciador |

Isso iniciará o programa e você verá o menu inicial do jogo no console.

## Funcionamento da Interface do Programa

Ao iniciar o jogo você verá um menu com duas opções:

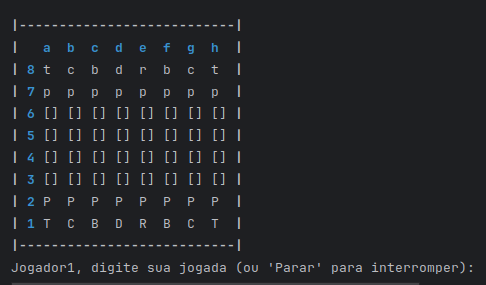
1. Iniciar novo Jogo: Selecionando esta opção, uma nova partida será iniciada.
2. Carregar Jogo: Permite carregar um jogo salvo anteriormente.  Ao selecionar esta opção será necessário informar o nome do arquivo no qual você salvou seu jogo anteriormente. O nome deve ser digitado corretamente. Por exemplo, se você salvou seu último jogo em um arquivo “jogadas.txt” você deverá digitar “jogadas.txt”

**Se você está jogando pela primeira vez**

Após executar e escolher a opção 1: iniciar novo jogo, será informado que está iniciando uma nova partida de Xadrez.

O programa pedirá para você informar o nome do Jogador 1, que será o jogador com as peças brancas e o nome do Jogador 2, que será o jogador com as peças pretas.

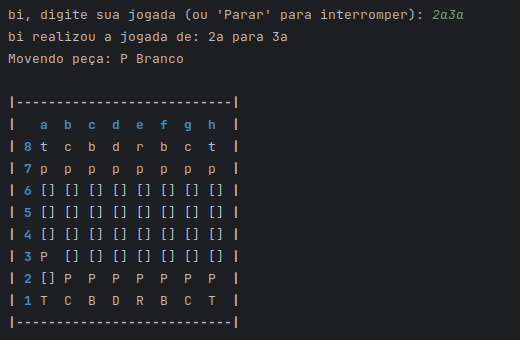
Após informar o nome dos jogadores, estaremos prontos para começar!

Quando o jogo finalmente começa, a primeira coisa que você verá é uma representação do tabuleiro. Os números em azul representam as linhas, e as letras em azul representam as colunas.

As peças brancas do Jogador 1, são representadas pelas letras maiúsculas na parte inferior do tabuleiro, e as peças pretas do Jogador 2 são representadas pelas letras em minúsculo no topo do tabuleiro.

Já as peças em si, são representadas pelas iniciais de seu nome, sendo elas:

* P = Peão
* T = Torre
* C = Cavalo
* B = Bispo
* D = Dama
* R = Rei



Será solicitado ao Jogador 1 para inserir a sua jogada. As jogadas são recebidas no formato *Linha Coluna Linha Coluna* exemplo 2a3a. Os primeiros dois valores são correspondentes a posição que a peça se encontra no momento da jogada, os seguintes valores são correspondentes a casa de destino da peça.

Então visualizando o tabuleiro ao lado, entrando como Jogador 1 a jogada 2a3a moveria o Peão na linha 2 coluna a, para a casa na linha 3.

A imagem acima mostra que ao realizar a jogada, o programa exibirá uma mensagem informando a jogada de uma origem para o destino, informa também a cor e a peça a ser movida.

**Jogadas Inválidas:**

Ao tentar realizar uma jogada inválida, o programa mostrará uma mensagem que a jogada que você está tentando realizar, é inválida e repetirá a mensagem para você digitar sua jogada. Entre os casos onde uma jogada será considerada inválida podemos citar:

* **Tentar movimentar uma peça em uma direção que não é permitida pela peça.** Por exemplo: uma Torre só pode se movimentar para frente e para o lado. Se você tentar movimentar uma Torre na diagonal, exibirá uma mensagem de jogada inválida.
* **Tentar mover uma peça que não pertence ao jogador atual.** Por exemplo, se o turno é do jogador preto e ele tenta movimentar uma peça do jogador branco.
* **Tentar capturar o próprio rei.**
* **Movimento inválido de captura.** Por exemplo, peões podem capturar apenas movimentos diagonais, mas se tentarem capturar em linha reta, a jogada será inválida.
* **Mover uma peça para uma posição ocupada por uma peça de mesma cor.** Exemplo: tentar mover uma peça branca para uma casa já ocupada

**Jogadas para fora do tabuleiro:**

Ao tentar mover uma peça para fora dos limites do tabuleiro, ou seja, além das colunas ‘a’ a ‘h’ e linhas 1 a 8. A movimentação fora do tabuleiro resultará em uma mensagem de erro, informando que a casa na qual você está tentando mover sua peça, não existe, e pedirá para o jogador digitar sua jogada novamente.

**Finalizando o jogo**

O jogo só para em dois casos: um dos jogadores sofrem xeque-mate ou você digita “parar” ao invés de realizar uma jogada.

Ao finalizar, o programa exibirá um menu perguntando se você deseja salvar esta partida, digitando 1 para sim, 2 para não. Caso você escolha salvar a partida, o programa pedirá para você informar o nome do arquivo no qual você deseja salvar.

Lembre-se de informar a extensão do aquivo! Por exemplo: *arquivo.txt*

Caso você escolha não salvar, o programa finalizará por completo.

# Sobre o projeto e as dificuldades encontradas

As informações a seguir são referente ao 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.
  + *Para tratar os erros encontrados reestruturamos as classes de forma que apenas os métodos acessados em outras classes fossem públicos.*
* 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.
  + *Para lidar com os possíveis erros de String para posições da classe casa, fizemos o devido casting das linhas para int e das colunas para char.*
* 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.
  + *Para tratar erros de jogadas inválidas ou fora do tabuleiro, jogamos uma exceção de IllegalArgumentException para gerar um aviso de jogada inválida ou fora do tabuleiro, evitando que o jogo quebre.*
* 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.
  + *Para evitar um erro onde por um acaso o oponente seja nulo, para evitar um erro ao verificarmos as condições de xeque e xeque-mate, jogamos uma exceção de IllegalArgumentException gerando um aviso. Evitando que o programa quebre.*

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

Durante o desenvolvimento do jogo, algumas funcionalidades não foram implementadas. São elas:

* **Empate:** o jogo não implementa as regras que podem resultar em empate, como afogamento, repetição de posição, insuficiência de material ou a regra dos 50 movimentos.
* **Roque:** a jogada especial de roque, que permite mover o rei e a torre simultaneamente, não foi incluída.
* **Promoção de Peão:** quando o peão atinge a oitava fileira, ele deve ser promovido para outra peça (geralmente uma dama). Esta funcionalidade também não foi implementada.

# Condições que podem causar erro

// por favor alguém escreve alguma coisa aqui eu não sei o que escrever KKKKK NÃO ACHEI ONDE PODE CAUSAR ERRO