Universidade Federal de São Carlos
Departamento de computação de Sorocaba

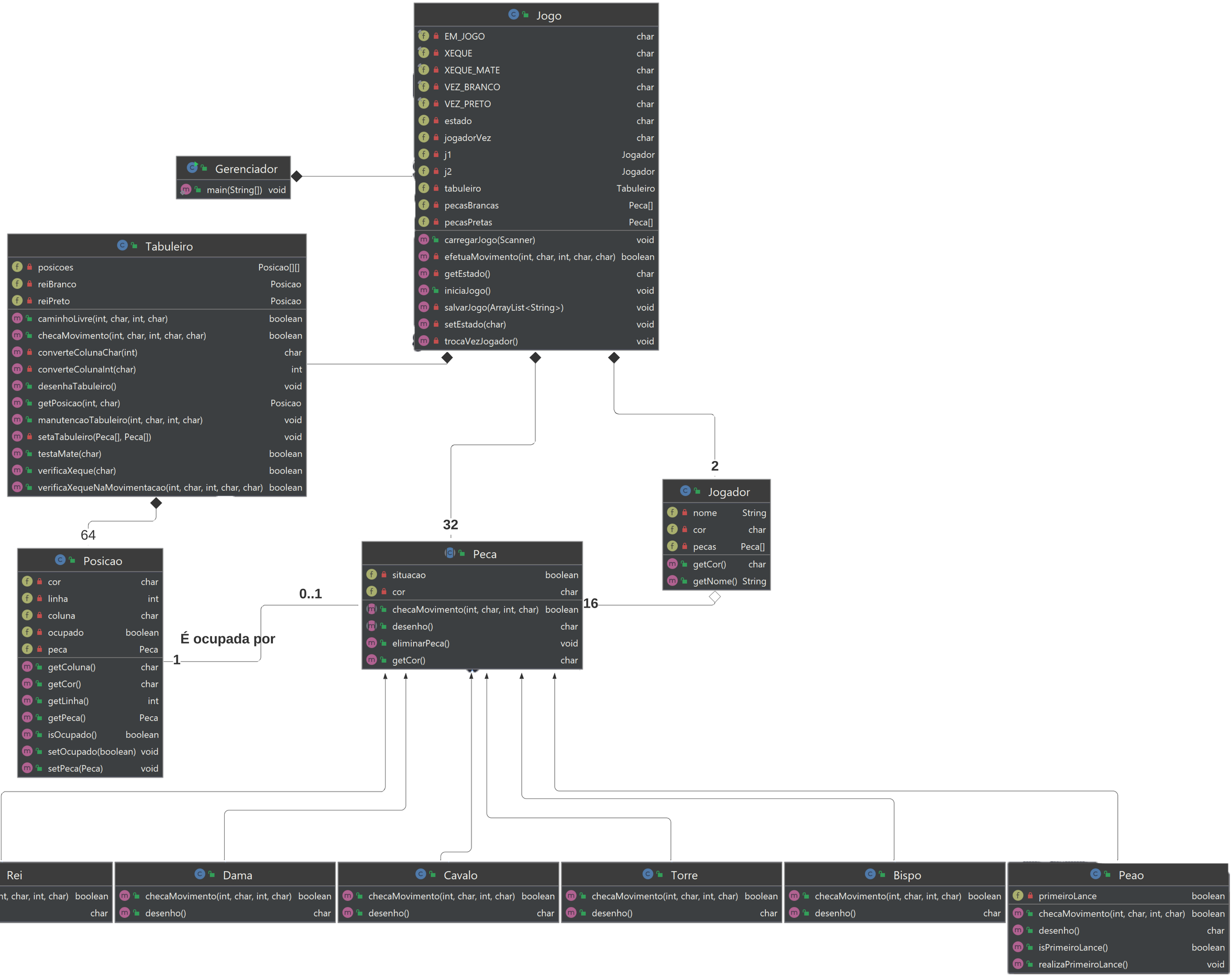
Programação Orientada a Objetos

Bruno dos Santos Ferraz - 796085
João Pedro Moro Bolognini - 792185

Relatório Final - Projetos 2 e 3

Sorocaba - SP
2021

Diagrama de Classes



Descrição das classes:

Gerenciador:

A classe Gerenciador se comunica com o usuário antes do jogo ser iniciado, fazendo o usuário escolher se quer um novo jogo ou carregar um jogo salvo, pedindo que o usuário insira os nomes dos jogadores ou o nome do arquivo onde está um jogo salvo para carregá-lo, dependendo da opção escolhida.Criando um jogo do zero ou carregando um jogo salvo de onde parou.

Jogo:

A classe Jogo contém métodos para gerenciamento do jogo, controlando tudo o que acontece no jogo. Também sabe o estado em que se encontra o jogo (em jogo, xeque ou xeque-mate). É a principal responsável pela comunicação com os utilizadores do jogo. Ela contém 2 jogadores e um tabuleiro.

Suas constantes são:

- EM_JOGO = ‘j‘, que indica que jogo está em curso.
- XEQUE = ’x’’ , que indica que o jogo está em mate.
- XEQUE_MATE = ‘m’, que indica que o jogo está em xeque-mate.
- VEZ_BRANCO = ‘b’, que indica a vez como sendo a do jogador branco.
- VEZ_PRETO = ‘p’, que indica a vez sendo como a do jogador preto.

Seus atributos são:

- estado, que indica se o jogo está em curso, se está em xeque ou em xeque-mate.
- jogadorVez, que indica de qual jogador é a vez.
- j1 e j2, objetos da classe Jogador, que são os dois jogadores do jogo.
- tabuleiro, que é o tabuleiro que o jogo vai usar.
- pecasBrancas, que representam as peças brancas.
- pecasPretas, que representam as peças pretas.

Seus métodos são:

- Jogo(), que é o construtor, que vai receber o nome dos dois jogadores e a cor do jogador 1, dessa forma arrumando o jogador 1 para aquele que vai ter a cor branca e o jogador 2 o outro, e define que a vez é do jogador da cor branca. Também cria e atribui as peças de cada jogador.
- iniciaJogo(), inicializa o jogo, criando o tabuleiro, recebendo as jogadas e imprime o tabuleiro a cada jogada até que ocorra o xeque-mate e salva um jogo a qualquer momento.

- efetuaMovimento(), que vai informar se a jogada recebida é válida ou não.
- trocaVezJogador(), que vai trocar a vez do jogador.
- getEstado(), que vai retornar o estado do jogo, em jogo, xeque ou xeque-mate.
- setEstado(), que vai trocar o estado do jogo.
- carregarJogo(), que vai carregar um determinado jogo de um arquivo salvo.
- salvarJogo(), salva o jogo atual em um arquivo.

Tabuleiro:

A classe Tabuleiro contém métodos para realizar a configuração inicial do tabuleiro, manutenção da configuração do tabuleiro a cada jogada e pelas checagens de adequação dos movimentos solicitados. Ela contém 64 posições.

Seus atributos são:
- posicoes, que é um vetor objeto da classe posicao, que representa as 64 casas do tabuleiro.
- ReiBranco, que é uma referência a posição do rei branco no tabuleiro.
- ReiPreto, que é uma referência a posição do rei preto no tabuleiro.

Seus métodos são:
- Tabuleiro(), que é o construtor da classe, que vai inicializar o tabuleiro criando as 64 posições dele, em 8 linhas e 8 colunas.
- setaTabuleiro(), que vai iniciar cada posição do tabuleiro com suas peças.
- desenhaTabuleiro(), que vai desenhar o tabuleiro na saída padrão dele, tendo 64 casas e suas linhas e colunas do lado.
- checaMovimento(), que verifica se a jogada passada como parâmetro é válida ou não para o tabuleiro.
- manutencaoTabuleiro(), que movimenta uma peça no tabuleiro, eliminando uma peça inimiga caso houver.
- caminhoLivre(), que verifica se um caminho está livre para a uma determinada peça efetuar o seu movimento.
- verificaXeque(), que verifica se o rei inimigo se encontra em xeque.
- verificaXequeNaMovimentacao(), que verifica se na movimentação do jogador, seu rei não vai entrar em xeque.
- testaMate(), que verifica se o jogador está em xeque-mate.
- getPosicao(), que retorna uma posição do tabuleiro.
- converteColunaInt(), que converte uma coluna de char para int.
- converteColunaChar(), que converte uma coluna de int para char.

Jogador:

A classe Jogador contém métodos para definir o nome e cor de cada jogador.

Seus atributos são:

- nome, que indica qual o nome do jogador.
- cor, que indica qual a cor das peças do jogador.
- pecas, que indica as peças do jogador.

Seu métodos são:

- Jogador(), que é o construtor, que define o nome, cor e o conjunto de peças do jogador.
- getNome(), que retorna o nome do jogador.
- getCor(), que retorna a cor do jogador.

Posição:

A classe Posicao contém métodos para definir a cor e coordenadas de cada posição. Também é responsável por manter a situação de cada posição, que pode ser vazia ou ocupada por uma peça, sabendo que peça a ocupa.

Seus atributos:

- cor, que indica qual a cor da posição, sendo branca ou preta.
- linha, que indica qual a linha daquela posição.
- coluna, que indica qual a coluna daquela posição.
- ocupado, que indica se há ou não há peça na posição.
- peca, que indica qual a peça está na posição se ela estiver ocupada.

Seus métodos são:

- Posicao(), que é o construtor, que vai definir a linha, a coluna, a cor, se está ou não ocupada e se houver peça, qual que está na posição.
- isOcupado(), que retorna true caso a posição esteja ocupada.
- setOcupado(), que modifica a situação da posição, se está ocupada ou não.
- getPeca(), que retorna qual peça está na posição.
- setPeca(), que modifica a peça na posição e modifica a situação da posição.
- getCor(), que retorna qual a cor da posição do tabuleiro.
- getLinha(), que retorna a linha da posição.,
- getColuna(), que retorna a coluna da posição.

Peca:

A classe abstrata Peca contém métodos para definir a cor e o desenho de cada peça. Também é responsável por manter e trocar a situação de cada peça, ou seja, se ela se encontra em jogo ou capturada.

Seus atributos são:

- situação, que indica se a peça está em jogo ou já foi capturada.
- cor, que indica a cor da peça, sendo branca ou preta.

Seus métodos são:

- checaMovimento(), método abstrato implementado por cada peça específica.
- desenho(), método abstrato implementado por cada peça específica.
- eliminarPeca(), elimina a peça do jogo, deixando sua situação como false, capturada.
- getCor(), que retorna a cor da peça.

Rei:

A classe Rei estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso uma casa em qualquer direção.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça rei.
- desenho(), que retorna a letra que representa a peça no jogo.

.**Dama:**

A classe Dama estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso várias casas na mesma coluna, linha ou diagonal.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça dama.
- desenho(), que retorna a letra que representa a peça no jogo.

Cavalo:

A classe Cavalo estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso duas linhas e uma coluna ou uma linha e duas colunas.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça cavalo.
- desenho(), que retorna a letra que representa a peça no jogo.

Bispo:

A classe Bispo estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso várias casas nas diagonais.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça bispo.
- desenho(), que retorna a letra que representa a peça no jogo.

Torre:

A classe Torre estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso várias casas na mesma coluna ou linha.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça torre.
- desenho(), que retorna a letra que representa a peça no jogo.

Peao:

A classe Peao estende a classe Peca e contém métodos para realizar a representação da peça na tela e checagem dos seus movimentos, neste caso uma casa ou duas casas para frente ou uma casa para diagonal caso tenha uma peça na sua diagonal

Seus atributos são:

- primeiroLance, que indica se a peça está no primeiro lance.

Seus métodos são:

- checaMovimento(), que checa se o movimento é válido ou não para a peça peão.
- desenho(), que retorna a letra que representa a peça no jogo.
- realizaPrimeiroLance(), que define a situação do primeiro lance como false.
- isPrimeiroLance(), verifica se o peão ja realizou seu primeiro movimento na partida.

Instruções para utilização

Requerimentos:

- java 11.0.12 2021-07-20 LTS ou superior
- Powershell, disponível em https://www.microsoft.com/store/productId/9MZ1SNWT0N5D ou um terminal linux.

É possível utilizar o Command Prompt do Windows, mas o tabuleiro pode apresentar inconsistências no seu alinhamento.

Como exemplo utilizamos o PowerShell:

1. Abra o PowerShell e navegue até a pasta do projeto através do comando cd:
   > cd .\Desktop\Xadrez\
2. Compile o programa:
   > javac *.java
3. Execute o programa:
   > java Gerenciador

Ao iniciar o programa, ele vai perguntar se quer começar um jogo do zero ou carregar um jogo salvo. Esse jogo salvo tem as instruções salvas em um arquivo.txt como o nome dos jogadores, o estado em que parou o jogo e as movimentações realizadas até o momento que foi salvo.
No tabuleiro, as peças em letra maiúscula representam as peças de cor branca e as peças em letra minúscula representam as peças de cor preta. O ¹ representa as casas brancas e o ² representa as casas pretas. As coordenadas do tabuleiro podem ser visualizadas em suas bordas.
O jogo funciona da seguinte forma, ao compilar e executar o jogo, ele vai pedir o nome do jogador das peças brancas e o nome do jogador das peças pretas, insira cada um, depois ele vai solicitar que quem esteja usando o programa insira as jogadas, que devem ser inseridas no seguinte formato : colunaOrigem linhaOrigem colunaDestino linhaDestino, sem espaços. As jogadas serão inseridas até ocorrer um xeque-mate no

jogo. Se o estado do jogo estiver em xeque, ele não vai deixar o jogador que está em xeque efetuar um movimento que não tire ele do xeque, sendo obrigatório ele escapar do xeque, se não tiver como escapar, será xeque-mate. A cada jogada ele vai imprimir o tabuleiro na tela, com as peças atualizadas após o movimento. Peças que forem eliminadas, serão retiradas do tabuleiro. Ao acabar o jogo, ele informará que ocorreu um xeque-mate e informará qual jogador venceu a partida.Para salvar um jogo, quando for inserir a movimentação de uma peça digite “salvar” .

Demonstração da interface

Tela inicial do jogo com as duas opções disponíveis:

Solicitando o nome dos dois jogadores e imprimindo o tabuleiro na sua configuração inicial (opção 1 - Novo Jogo):

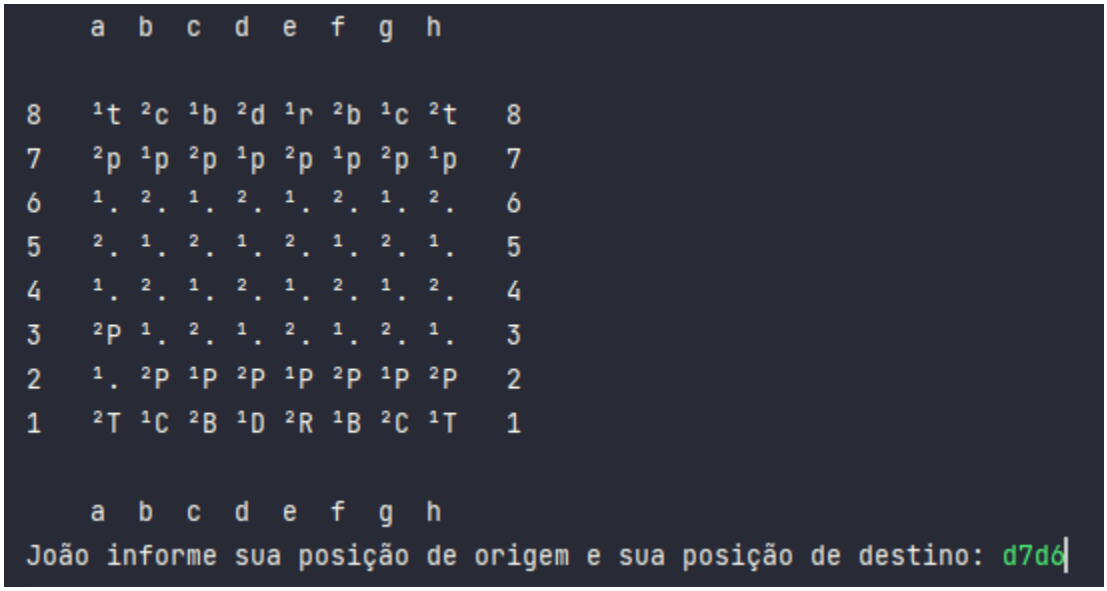
Carregando um jogo existente, imprimindo o tabuleiro na última jogada executada e solicitando a jogada para o jogador correspondente (opção 2 - Carregar jogo):

```
Digite o nome do arquivo que deseja carregar:
configjogo

---Instruções gerais---
Para salvar o jogo a qualquer momento digite "salvar".
O formato de entrada deve ser colunaOrigem, linhaOrigem, colunaDestino, linhaDestino e deve ser inserido nessa ordem, sem espaço entre as casas.
Por exemplo: a2a3

Jogador 1 de cor branca: Bruno
Jogador 2 de cor preta: João

O rei branco está em xeque
    a  b  c  d  e  f  g  h

8   ¹. ². ¹. ². ¹r ². ¹. ²t   8
7   ². ¹p ². ¹. ². ¹p ². ¹.   7
6   ¹. ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹b ². ¹. ². ¹. ². ¹P   5
4   ¹. ²b ¹. ². ¹. ². ¹. ²D   4
3   ². ¹. ². ¹. ²P ¹. ². ¹.   3
2   ¹t ². ¹P ². ¹. ²P ¹. ². ¹.   2
1   ². ¹. ². ¹. ²R ¹. ². ¹.   1

    a  b  c  d  e  f  g  h
Bruno informe sua posição de origem e sua posição de destino:
```

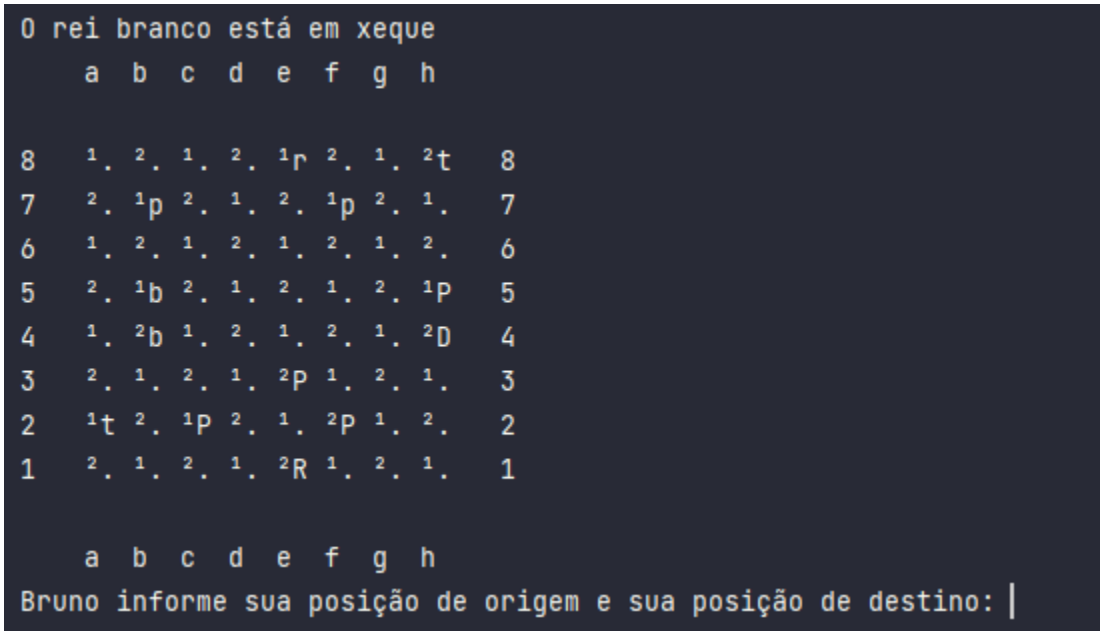
Solicitando ao jogador das peças brancas para informar sua movimentação. Observe como a entrada é inserida.

```
    a  b  c  d  e  f  g  h

8   ¹t ²c ¹b ²d ¹r ²b ¹c ²t   8
7   ²p ¹p ²p ¹p ²p ¹p ²p ¹p   7
6   ¹. ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹. ². ¹. ². ¹. ². ¹.   5
4   ¹. ². ¹. ². ¹. ². ¹. ².   4
3   ². ¹. ². ¹. ². ¹. ². ¹.   3
2   ¹P ²P ¹P ²P ¹P ²P ¹P ²P   2
1   ²T ¹C ²B ¹D ²R ¹B ²C ¹T   1

    a  b  c  d  e  f  g  h
Bruno informe sua posição de origem e sua posição de destino: a2a3
```

Solicitando ao jogador das peças pretas para informar sua movimentação. Observe como a entrada é inserida.



Causando um xeque, tentando fazer um movimento que não o tira do xeque e fazendo um que o tira, podemos observar que ele informa que a jogada é inválida na jogada que não tira.



```
Bruno informe sua posição de origem e sua posição de destino: e1f1
Movimento inválido!
O rei branco está em xeque
    a  b  c  d  e  f  g  h

8   ¹. ². ¹. ². ¹r ². ¹. ²t   8
7   ². ¹p ². ¹. ². ¹p ². ¹.   7
6   ¹. ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹b ². ¹. ². ¹. ². ¹P   5
4   ¹. ²b ¹. ². ¹. ². ¹. ²D   4
3   ². ¹. ². ¹. ²P ¹. ². ¹.   3
2   ¹t ². ¹P ². ¹. ²P ¹. ².   2
1   ². ¹. ². ¹. ²R ¹. ². ¹.   1
```

```
Bruno informe sua posição de origem e sua posição de destino: e1d1
Movimento válido!
    a  b  c  d  e  f  g  h

8   ¹. ². ¹. ². ¹r ². ¹. ²t   8
7   ². ¹p ². ¹. ². ¹p ². ¹.   7
6   ¹. ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹b ². ¹. ². ¹. ². ¹P   5
4   ¹. ²b ¹. ². ¹. ². ¹. ²D   4
3   ². ¹. ². ¹. ²P ¹. ². ¹.   3
2   ¹t ². ¹P ². ¹. ²P ¹. ².   2
1   ². ¹. ². ¹R ². ¹. ². ¹.   1
```

Demonstração da interface ao chegar em um xeque-mate:

```
João informe sua posição de origem e sua posição de destino: a2a1
Movimento válido!
    a  b  c  d  e  f  g  h

8   ¹. ². ¹. ². ¹r ². ¹. ²t   8
7   ². ¹p ². ¹. ². ¹p ². ¹.   7
6   ¹. ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹b ². ¹. ². ¹. ². ¹p   5
4   ¹. ²b ¹. ². ¹. ². ¹. ²D   4
3   ². ¹. ². ¹. ²P ¹. ². ¹.   3
2   ¹. ². ¹P ². ¹. ²P ¹. ².   2
1   ²t ¹. ². ¹R ². ¹. ². ¹.   1

    a  b  c  d  e  f  g  h
-- Xeque-Mate --
João de peças pretas venceu o jogo!
```

Demonstração de um jogo sendo salvo:

```
    a  b  c  d  e  f  g  h

8   ¹t ²c ¹b ²d ¹r ²b ¹c ²t   8
7   ². ¹p ²p ¹p ²p ¹p ²p ¹p   7
6   ¹p ². ¹. ². ¹. ². ¹. ².   6
5   ². ¹. ². ¹. ². ¹. ². ¹.   5
4   ¹. ². ¹. ². ¹. ². ¹. ².   4
3   ²P ¹. ². ¹. ². ¹. ². ¹.   3
2   ¹. ²P ¹P ²P ¹P ²P ¹P ²P   2
1   ²T ¹C ²B ¹D ²R ¹B ²C ¹T   1

    a  b  c  d  e  f  g  h
Bruno informe sua posição de origem e sua posição de destino: salvar
Digite o nome do arquivo: jogo1
Gravado com sucesso no arquivo jogo1.txt
```

Limitações e problemas

O programa não suporta as jogadas especiais, tais como roque, captura por passagem e promoção de peças. Não foram encontrados outros problemas que causassem funcionamentos inesperados no programa.

Exceções e tratamentos

Classe Peca:

Condição de erro:

1. Uma peça não pode ser criada com uma cor diferente de ‘b’ ou ‘p’.

Tratamento:

1. No construtor da classe peça é verificado se a cor é branco ou preto, lançando uma exceção informando qual peça está com a cor inválida.

Todas as classes filhas desta classe adicionam a exceção na assinatura do construtor.

Classe Jogador:

Condições de erro:

1. Um jogador não pode ser criado com uma cor diferente de ‘b’ ou ‘p’.
2. Um jogador não pode ser criado com uma cor diferente das suas peças atribuídas.

Tratamento:

1. No construtor da classe Jogador é verificado se a cor é branco ou preto, lançando uma exceção informando qual jogador e a respectiva cor inserida é inválida.
2. No construtor da classe Jogador é verificado se a cor é diferente da cor das suas peças atribuídas , lançando uma exceção informando que a cor das peças é diferente da cor do jogador.

Classe Posição:

Condições de erro:

1. Uma posição não pode ser criada com uma linha menor que 1 ou maior que 8. Uma posição não pode ser criada com uma coluna menor que ‘a’ ou maior que ‘h’.

Tratamento:

1. No construtor da classe posição é verificado se a linha é menor que 1 ou maior que 8, lançando uma exceção informando que a linha é inválida para posição.
2. No construtor da classe posição é verificado se a coluna é menor que ‘a’ ou maior que ’h’, lançando uma exceção informando que a coluna é inválida para posição.

Classe Tabuleiro:

Condições de erro:

1. A classe tabuleiro cria posições em seu construtor e em alguns métodos, podendo causar exceções da classe posição.

Tratamento:

1. O construtor e todos os métodos que criam uma determinada posição adicionam a exceção da classe posição em suas assinaturas.

Classe Jogo:

Condições de erro:

1. A classe jogo cria objetos e utiliza métodos da classe Peca, Jogador e Tabuleiro, podendo causar exceções das respectivas classes.
2. Ao salvar um arquivo pode ocorrer um erro inesperado, por exemplo, estar corrompido.
3. Ao solicitar uma entrada para fazer a movimentação, se não for válida causará uma exceção.

Tratamento:

1. O construtor e todos os métodos que criam e utilizam os métodos da classe Peca, Jogador e Tabuleiro adicionam a exceção em suas assinaturas.
2. A exceção de entrada e saída é tratada no próprio método, informando que houve um erro ao salvar o jogo.
3. A exceção é tratada no próprio método, solicitando a inserção de uma nova jogada.

Classe Gerenciador:

Condições de erro:

1. Ao carregar um arquivo de um determinado jogo ele pode não ser encontrado.

2. Ao criar um novo jogo todas as exceções que foram lançadas nesta classe são tratadas na classe gerenciador, informando a devida mensagem de erro.

Tratamento:

1. A exceção de arquivo não encontrado é tratada no próprio método, informando que o arquivo não foi encontrado.
2. A exceção de entrada e saída é tratada no próprio método, informando que houve um erro ao salvar o jogo.