Universidade Federal de São Carlos - Campus Sorocaba

Ciência da Computação

Programação Orientada a Objetos

Jogo de Xadrez

Fase 1 - Data de entrega: 30/10/2020

Prof. Dra. Katti Faceli

Nome:Felipe Augusto Vale Amorim RA: 743530

Sorocaba / SP

2020

Sumário

1. Introdução

Foi proposto pela Professora Dra. Katti Faceli, docente da disciplina de Programação Orientada a Objetos, o trabalho responsável pelo gerenciamento de uma partida de Xadrez. A proposta seria desenvolver e implementar uma solução computacional em Java que seja responsável pelo controle de todas as atividades e movimentos que acontecem em um jogo de xadrez, de acordo com as regras para cada peça.

PALAVRAS CHAVES: Xadrez, Peça, Tabuleiro

2. Descrição das Classes

Nesta seção é apresentada a descrição de todas as classes do projeto. Sua representação gráfica (diagrama de classes) pode ser encontrada no arquivo “Diagrama de Classes.png”.

O sistema desenvolvido está um pouco diferente da representação gráfica, porém isso irá acontecer apenas nessa fase, já que não possuímos todas as classes implementadas como *Peca*, *Jogador* e *Gerenciador*.

2.1 Classe Teste

A classe *Teste* é responsável pela impressão de um menu para que o usuário possa informar as peças e seus movimentos. Aqui também é criado o objeto de tipo *Jogo.*

Possui os seguintes atributos:

- não possui atributos.

Possui os seguintes métodos:

- main(): cria um menu selecionável ao usuário para informar qual o tipo de peça específica que será testado e chama os métodos que testam as peças. Também imprime ao usuário informando se o movimento da peça é válido ou não.

2.2 Classe Jogo

A classe *Jogo* é responsável por iniciar a partida de Xadrez e gerenciar a partida.

Possui os seguintes atributos:

- tabuleiro: indica o tabuleiro da partida;
- estado: indica o estado da partida, podendo ser “Início”, “Xeque” ou “Xeque-Mate”;
- vezJogador: indica de qual jogador é a vez, podendo receber 0 (branco) ou 1 (preto).

Possui os seguintes métodos:

- Jogo(): construtor responsável por criar o objeto de tipo *Jogo*;
- testePecas(): responsável por receber as informações da peça e movimentação no tabuleiro que o usuário informou e enviar para as funções que testam a movimentação da peça (apenas nessa fase).

2.3 Classe Tabuleiro

A classe *Tabuleiro* é responsável por setar as configurações iniciais do tabuleiro na partida e pela manutenção e checagem das peças.

Possui os seguintes atributos:

- mesa: indica as 64 posições do tabuleiro.

Possui os seguintes métodos:

- Tabuleiro(): construtor responsável por criar o objeto de tipo *Tabuleiro*. Aqui é feito as configurações iniciais do tabuleiro, antes da partida começar;
- getPosicao(): responsável por retornar o objeto de tipo *Posição* dentro do tabuleiro, de acordo com as coordenadas informadas;
- desenho(): responsável por imprimir em tela as configurações atuais do tabuleiro;
- checaMovimento(): responsável por verificar a movimentação da peça dentro do tabuleiro e informar se a movimentação é válida ao usuário.

2.4 Classe Posição

A classe *Posição* é responsável por armazenar as informações de uma posição dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a posição é branca ou branca;
- linha: indica a coordenada linha da posição;
- coluna: indica a coordenada coluna da posição;
- ocupante: indica qual peça está ocupando a posição, podendo ser vazia.

Possui os seguintes métodos:

- isCor(): responsável por retornar qual é a cor da posição no tabuleiro;
- setCor(): responsável por definir a cor da posição no tabuleiro;
- setLinha(): responsável por definir a coordenada linha da posição no tabuleiro;
- setColuna(): responsável por definir a coordenada coluna da posição no tabuleiro;
- setOcupante(): responsável por definir a peça que ocupa a posição no tabuleiro.

2.5 Classe Peao

A classe *Peao* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;
- inGame: indica se a peça está em jogo ou não;
- primeiraJogada: indica se é a primeira movimentação da peça ou não. Isso é necessário, pois o peão, em caso de primeira jogada, tem uma movimentação diferente da padrão.

Possui os seguintes métodos:

- Peao(): responsável por construir o objeto de tipo *Peao*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;

- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

2.6 Classe Bispo

A classe *Bispo* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;
- inGame: indica se a peça está em jogo ou não;

Possui os seguintes métodos:

- Bispo(): responsável por construir o objeto de tipo *Bispo*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;
- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

2.7 Classe Torre

A classe *Torre* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;
- inGame: indica se a peça está em jogo ou não;

Possui os seguintes métodos:

- Torre(): responsável por construir o objeto de tipo *Torre*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;
- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

2.8 Classe Cavalo

A classe *Cavalo* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;
- inGame: indica se a peça está em jogo ou não;

Possui os seguintes métodos:

- Cavalo(): responsável por construir o objeto de tipo *Cavalo*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;
- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

2.9 Classe Dama

A classe *Dama* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;
- inGame: indica se a peça está em jogo ou não;

Possui os seguintes métodos:

- Dama(): responsável por construir o objeto de tipo *Dama*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;
- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

2.10 Classe Rei

A classe *Rei* é responsável por construir a peça, fazer suas verificações para movimentação e retornar o desenho da peça que será inserido dentro do tabuleiro.

Possui os seguintes atributos:

- cor: indica se a peça é branca ou branca;

- inGame: indica se a peça está em jogo ou não;

Possui os seguintes métodos:

- Rei(): responsável por construir o objeto de tipo *Rei*;
- getCor(): responsável por retornar a cor da peça;
- desenho(): responsável por retornar o “desenho” (peça preta: letra maiúscula, peça branca: letra minúscula) da peça no tabuleiro ;
- checaMovimento(): responsável por verificar se o movimento da peça é válido de acordo com as regras do jogo de xadrez.

3. Condições que podem causar erros

Durante o desenvolvimento da aplicação, pude notar três possíveis erros:

- O primeiro erro poderia ser causado no momento que o usuário informa ao sistema, uma entrada que viola as dimensões do tabuleiro. Esta condição foi tratada já nesta fase;
- O segundo erro poderia ser causado no momento que o usuário informa ao sistema, uma entrada inválida de acordo com a variada definida no sistema, como por exemplo, inserir um valor de coluna em uma entrada que espera uma linha. Esta condição não foi tratada nesta fase;
- .O terceiro erro poderia ser causado no momento que o usuário tenta realizar uma movimentação inválida de acordo com a peça selecionada. Esta condição foi tratada nesta fase.

4. Testes realizados

Os testes informados abaixo foram feitos de acordo com a ordem de execução com que o Jogo de Xadrez é construído, seguindo o diagrama de classes apresentado anteriormente.

Sendo assim, a primeira coisa a ser construída é um objeto *Jogo*. No construtor da classe *Jogo* é construído um objeto *Tabuleiro*, que em seu construtor, é populado com 64 objetos *Posição*.

A entrada de dados referentes a origem e destino das peças são informados na classe *Teste*, através da classe *Scanner()*, e os objetos das peças específicas são construídos no método “checaMovimento*(char peca, int cor, int linhaOrigem, String*

colunaOrigem, int linhaDestino, String colunaDestino)" da classe *Tabuleiro* (nesta fase, apenas).

As informações chegam neste método através de um método de teste construído apenas para essa fase na classe *Jogo*, que é "*testePecas(char peca, int linhaOrigem, String colunaOrigem, int linhaDestino, String colunaDestino)"*. Este método retorna se o movimento é válido ou não da peça dentro do tabuleiro (true ou false).

Os testes realizados podem ser vistos abaixo:

Teste 1: Construir o objeto de tipo *Jogo*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Jogo*.

Objeto: j.

Método testado: Jogo().

Valores dos parâmetros:

Valor de retorno:

Tela:

Teste 2: Construir o objeto de tipo *Tabuleiro*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Tabuleiro*.

Objeto: tabuleiro.

Método testado: Tabuleiro().

Valores dos parâmetros:

Valor de retorno:

Tela:

Teste 3: Construir os objetos de tipo *Posição*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Posição*.

Objeto: mesa.

Método testado: Posicao().

Valores dos parâmetros:

Valor de retorno:

Tela:

Teste 4: Popular o objeto de tipo *Tabuleiro* com os objetos de tipo *Posição* (linha).

Objetivo do teste: O objetivo do teste é popular o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*. Neste teste é realizada as configurações iniciais do tabuleiro para o início da partida.

Objeto: mesa.

Método testado: setLinha(int linha).

Valores dos parâmetros: setLinha(i), onde i é iniciado com valor 8 e diminui até possuir valor maior que 0.

Valor de retorno:

Tela:

Teste 5: Popular o objeto de tipo *Tabuleiro* com os objetos de tipo *Posição* (coluna).

Objetivo do teste: O objetivo do teste é popular o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*. Neste teste é realizada as configurações iniciais do tabuleiro para o início da partida.

Objeto: mesa.

Método testado: setColuna(String coluna).

Valores dos parâmetros: setColuna(coluna), onde coluna é um valor entre as letras “A” e “H”.

Valor de retorno:

Tela:

Teste 6: Popular o objeto de tipo *Tabuleiro* com os objetos de tipo *Posição* (ocupante).

Objetivo do teste: O objetivo do teste é popular o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*. Neste teste é realizada as configurações iniciais do tabuleiro para o início da partida. Nesta fase, o método irá apenas receber uma string vazia.

Objeto: mesa.

Método testado: setOcupante(String ocupante).

Valores dos parâmetros: setOcupante(“”), onde “” representa que a posição ainda não foi ocupada por nenhuma peça.

Valor de retorno:

Tela:

Teste 7: Popular o objeto de tipo *Tabuleiro* com os objetos de tipo *Posição* (cor branca).

Objetivo do teste: O objetivo do teste é popular o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*. Neste teste é realizada as configurações iniciais do tabuleiro para o início da partida.

Objeto: mesa.

Método testado: setCor(int cor).

Valores dos parâmetros: setCor(0), onde 0 representa que a posição possui cor branca.

Valor de retorno:

Tela:

Teste 8: Popular o objeto de tipo *Tabuleiro* com os objetos de tipo *Posição* (cor preta).

Objetivo do teste: O objetivo do teste é popular o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*. Neste teste é realizada as configurações iniciais do tabuleiro para o início da partida.

Objeto: mesa.

Método testado: setCor(int cor).

Valores dos parâmetros: setCor(1), onde 1 representa que a posição possui cor preta.

Valor de retorno:

Tela:

Teste 9: Desenhar o tabuleiro.

Objetivo do teste: O objetivo do teste é imprimir para o usuário o objeto de tipo *Tabuleiro* com os 64 objetos de tipo *Posição*.

Objeto: tabuleiro.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno:

Tela:

```
    A B C D E F G H
8   0 1 0 1 0 1 0 1
7   1 0 1 0 1 0 1 0
6   0 1 0 1 0 1 0 1
5   1 0 1 0 1 0 1 0
4   0 1 0 1 0 1 0 1
3   1 0 1 0 1 0 1 0
2   0 1 0 1 0 1 0 1
1   1 0 1 0 1 0 1 0
```

Teste 10: Recuperar os objetos de tipo *Posição* do tabuleiro.

Objetivo do teste: O objetivo do teste é recuperar do objeto de tipo *Tabuleiro*, os objetos de tipo *Posição*.

Objeto: mesa.

Método testado: getPosicao(int linha, int coluna).

Valores dos parâmetros: getPosicao(i-1, j), onde i é um valor que inicia em 8 e diminui até ser maior que 0 e j é um valor que inicia em 0 e é interado até continuar sendo menor que 8.

Valor de retorno: this.mesa[linha][coluna], onde linha é um valor que inicia em 8 e diminui até ser maior que 0 e coluna é um valor que inicia em 0 e é interado até continuar sendo menor que 8.

Tela:

Teste 11: Recuperar as cores das posições do tabuleiro.

Objetivo do teste: O objetivo do teste é recuperar do objeto de tipo *Tabuleiro*, as cores dos objetos de tipo *Posição*.

Objeto: mesa.

Método testado: isCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 12: Construir o objeto de tipo *Peao*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Peao*.

Objeto: novoP.

Método testado: Peao(int cor).

Valores dos parâmetros: Peao(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 13: Desenhar peça peão.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novoP.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “d” para peças brancas, “D” para peças pretas.

Tela:

Teste 14: Retornar cor da peça peão.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novoP.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 15: Movimento para frente, do peão, em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando em uma casa para frente, o que é válido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “c”).

Valor de retorno: true

Tela: Testando Peao

Movimento válido! Sua posição atual é: (5, C)

Teste 16: Movimento para frente, do peão em duas casas, caso seja a primeira jogada.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando em duas casas para frente, caso seja a primeira jogada, o que é válido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “c”).

Valor de retorno: true

Tela: Testando Peao

Movimento válido! Sua posição atual é: (6, C)

Teste 17: Movimento para a diagonal da direita e para frente, do peão em uma casa, em caso de captura.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando em uma casa para a diagonal, em caso de captura, o que é válido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: true

Tela: Testando Peao

Movimento válido! Sua posição atual é: (5, D)

Teste 18: Movimento para a diagonal da esquerda e para frente, do peão em uma casa, em caso de captura.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando em uma casa para a diagonal, em caso de captura, o que é válido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “b”).

Valor de retorno: true

Tela: Testando Peao

Movimento válido! Sua posição atual é: (5, B)

Teste 19: Movimento para a direita, do peão.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando para a direita, o que é inválido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “d”).

Valor de retorno: false

Tela: Testando Peao

Movimento inválido! Sua posição atual é: (4, C)

Teste 20: Movimento para a esquerda, do peão.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando para a esquerda, o que é inválido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “b”).

Valor de retorno: false

Tela: Testando Peao

Movimento inválido! Sua posição atual é: (4, C)

Teste 21: Movimento para trás, do peão.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando para trás, o que é inválido.

Objeto: novoP.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “c”).

Valor de retorno: false

Tela: Testando Peao

Movimento inválido! Sua posição atual é: (4, C)

Teste 22: Construir o objeto de tipo *Bispo*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Bispo*.

Objeto: novoB.

Método testado: Bispo(int cor).

Valores dos parâmetros: Bispo(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 23: Desenhar peça bispo.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novoB.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “b” para peças brancas, “B” para peças pretas.

Tela:

Teste 24: Retornar cor da peça bispo.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novoB.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 25: Movimento para diagonal da direita e para frente (independentemente do número de casas), do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para a diagonal da direita, o que é válido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: true

Tela: Testando Bispo

Movimento válido! Sua posição atual é: (5, D)

Teste 26: Movimento para diagonal da esquerda e para frente (independentemente do número de casas), do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para a diagonal da esquerda, o que é válido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “b”).

Valor de retorno: true

Tela: Testando Bispo

Movimento válido! Sua posição atual é: (5, B)

Teste 27: Movimento para diagonal da direita e para trás (independentemente do número de casas), do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para a diagonal da direita, o que é válido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “d”).

Valor de retorno: true

Tela: Testando Bispo

Movimento válido! Sua posição atual é: (3, D)

Teste 28: Movimento para diagonal da esquerda e para trás (independentemente do número de casas), do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para a diagonal da esquerda, o que é válido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “b”).

Valor de retorno: true

Tela: Testando Bispo

Movimento válido! Sua posição atual é: (3, B)

Teste 29: Movimento para frente, do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para frente, o que é inválido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “c”).

Valor de retorno: false

Tela: Testando Bispo

Movimento inválido! Sua posição atual é: (4, C)

Teste 30: Movimento para trás, do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para trás, o que é inválido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “c”).

Valor de retorno: false

Tela: Testando Bispo

Movimento inválido! Sua posição atual é: (4, C)

Teste 31: Movimento para direita, do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para direita, o que é inválido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “d”).

Valor de retorno: false

Tela: Testando Bispo

Movimento inválido! Sua posição atual é: (4, C)

Teste 32: Movimento para esquerda, do bispo.

Objetivo do teste: O objetivo do teste é verificar se o bispo está andando para esquerda, o que é inválido.

Objeto: novoB.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “b”).

Valor de retorno: false

Tela: Testando Bispo

Movimento inválido! Sua posição atual é: (4, C)

Teste 33: Construir o objeto de tipo *Torre*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Torre*.

Objeto: novaT.

Método testado: Torre(int cor).

Valores dos parâmetros: Torre(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 34: Desenhar peça torre.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novaT.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “t” para peças brancas, “T” para peças pretas.

Tela:

Teste 35: Retornar cor da peça torre.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novaT.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 36: Movimento para frente (independentemente do número de casas), da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para a frente, o que é válido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “c”).

Valor de retorno: true

Tela: Testando Torre

Movimento válido! Sua posição atual é: (6, C)

Teste 37: Movimento para trás (independentemente do número de casas), da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para trás, o que é válido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “c”).

Valor de retorno: true

Tela: Testando Torre

Movimento válido! Sua posição atual é: (2, C)

Teste 38: Movimento para direita (independentemente do número de casas), da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para direita, o que é válido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “e”).

Valor de retorno: true

Tela: Testando Torre

Movimento válido! Sua posição atual é: (4, E)

Teste 39: Movimento para esquerda (independentemente do número de casas), da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para esquerda, o que é válido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “a”).

Valor de retorno: true

Tela: Testando Torre

Movimento válido! Sua posição atual é: (4, A)

Teste 40: Movimento para diagonal da direita e para frente, da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para a diagonal da direita, o que é inválido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: false

Tela: Testando Torre

Movimento inválido! Sua posição atual é: (4, C)

Teste 41: Movimento para diagonal da esquerda e para frente, da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para a diagonal da esquerda, o que é inválido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “b”).

Valor de retorno: false

Tela: Testando Torre

Movimento inválido! Sua posição atual é: (4, C)

Teste 42: Movimento para diagonal da direita e para trás, da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para a diagonal da direita, o que é inválido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “d”).

Valor de retorno: false

Tela: Testando Torre

Movimento inválido! Sua posição atual é: (4, C)

Teste 43: Movimento para diagonal da esquerda e para trás, da torre.

Objetivo do teste: O objetivo do teste é verificar se a torre está andando para a diagonal da esquerda, o que é inválido.

Objeto: novaT.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “b”).

Valor de retorno: false

Tela: Testando Torre

Movimento inválido! Sua posição atual é: (4, C)

Teste 44: Construir o objeto de tipo *Cavalo*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Cavalo*.

Objeto: novoC.

Método testado: Cavalo(int cor).

Valores dos parâmetros: Cavalo(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 45: Desenhar peça cavalo.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novoC.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “c” para peças brancas, “C” para peças pretas.

Tela:

Teste 46: Retornar cor da peça cavalo.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novoC.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 47: Movimento em L e para direita (inserindo na frente), do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L e para direita, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “d”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (6, D)

Teste 48: Movimento em L e para esquerda (inserindo na frente), do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L e para esquerda, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “a”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (5, A)

Teste 49: Movimento em L, para frente, e para direita, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L, para frente e para direita, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “d”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (6, D)

Teste 50: Movimento em L, para frente, e para esquerda, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L, para frente e para esquerda, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “b”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (6, B)

Teste 51: Movimento em L e para direita (inserindo atrás), do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L e para direita, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “e”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (3, E)

Teste 52: Movimento em L e para esquerda (inserindo atrás), do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L e para esquerda, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “a”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (3, A)

Teste 53: Movimento em L, para trás, e para direita, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L, para trás e para direita, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “d”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (2, D)

Teste 54: Movimento em L, para frente, e para esquerda, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está andando em L, para frente e para esquerda, o que é válido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “b”).

Valor de retorno: true

Tela: Testando Cavalo

Movimento válido! Sua posição atual é: (2, B)

Teste 55: Movimento para direita, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para direita, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “d”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 56: Movimento para esquerda, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para esquerda, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “b”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 57: Movimento para frente, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para frente, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “c”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 58: Movimento para trás, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para trás, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “c”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 59: Movimento para diagonal da direita, e para frente, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para a diagonal da direita, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 60: Movimento para diagonal da esquerda, e para frente, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para a diagonal da esquerda, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “c”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 61: Movimento para diagonal da direita, e para trás, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para a diagonal da direita, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “d”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 62: Movimento para diagonal da esquerda, e para trás, do cavalo.

Objetivo do teste: O objetivo do teste é verificar se o cavalo está indo para a diagonal da esquerda, o que é inválido.

Objeto: novoC.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “c”).

Valor de retorno: false

Tela: Testando Cavalo

Movimento inválido! Sua posição atual é: (4, C)

Teste 63: Construir o objeto de tipo *Dama*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Dama*.

Objeto: novaD.

Método testado: Dama(int cor).

Valores dos parâmetros: Dama(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 64: Desenhar peça dama.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novaD.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “d” para peças brancas, “D” para peças pretas.

Tela:

Teste 65: Retornar cor da peça dama.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novaD.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 66: Movimento para diagonal da direita e para frente (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para a diagonal da direita, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (5, D)

Teste 67: Movimento para diagonal da esquerda e para frente (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para a diagonal da esquerda, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “b”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (5, B)

Teste 68: Movimento para diagonal da direita e para trás (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para a diagonal da direita, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “d”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (3, D)

Teste 69: Movimento para diagonal da esquerda e para trás (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para a diagonal da esquerda, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “b”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (3, B)

Teste 70: Movimento para frente (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para a frente, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “c”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (6, C)

Teste 71: Movimento para trás (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para trás, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “c”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (2, C)

Teste 72: Movimento para direita (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para direita, o que é válido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “h”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (4, H)

Teste 73: Movimento para esquerda (independentemente do número de casas), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando para esquerda, o que é válido .

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “a”).

Valor de retorno: true

Tela: Testando Dama

Movimento válido! Sua posição atual é: (4, A)

Teste 74: Movimento em L (independentemente do número de casas) e para direita (inserindo na frente), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L e para direita, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “d”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 75: Movimento em L (independentemente do número de casas) e para esquerda (inserindo na frente), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L e para esquerda, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “a”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 76: Movimento em L (independentemente do número de casas), para frente, e para direita, da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L, para frente e para direita, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “d”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 77: Movimento em L (independentemente do número de casas), para frente, e para esquerda, da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L, para frente e para esquerda, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “b”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 78: Movimento em L (independentemente do número de casas) e para direita (inserindo atrás), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L e para direita, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “e”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 79: Movimento em L (independentemente do número de casas) e para esquerda (inserindo atrás), da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L e para esquerda, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “a”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 80: Movimento em L (independentemente do número de casas), para trás, e para direita, da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L, para trás e para direita, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “d”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 81: Movimento em L (independentemente do número de casas), para frente, e para esquerda, da dama.

Objetivo do teste: O objetivo do teste é verificar se a dama está andando em L, para frente e para esquerda, o que é inválido.

Objeto: novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “b”).

Valor de retorno: false

Tela: Testando Dama

Movimento inválido! Sua posição atual é: (4, C)

Teste 82: Construir o objeto de tipo *Rei*.

Objetivo do teste: O objetivo do teste é construir um objeto de tipo *Rei*.

Objeto: novoR.

Método testado: Rei(int cor).

Valores dos parâmetros: Rei(int cor), onde cor representa a cor da peça, 0 para branca e 1 para preta.

Valor de retorno:

Tela:

Teste 83: Desenhar peça rei.

Objetivo do teste: O objetivo do teste é desenhar a peça de acordo com sua cor, letras minúsculas são peças brancas enquanto letras maiúsculas são peças pretas.

Objeto: novoR.

Método testado: desenho().

Valores dos parâmetros:

Valor de retorno: “r” para peças brancas, “R” para peças pretas.

Tela:

Teste 84: Retornar cor da peça rei.

Objetivo do teste: O objetivo do teste é retornar qual é a cor de uma peça.

Objeto: novoR.

Método testado: getCor().

Valores dos parâmetros:

Valor de retorno: cor

Tela:

Teste 85: Movimento para frente, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para frente, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “c”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (5, C)

Teste 86: Movimento para trás, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para trás, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “c”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (3, C)

Teste 87: Movimento para a diagonal da direita e para frente, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para a diagonal da direita, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “d”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (5, D)

Teste 88: Movimento para a diagonal da esquerda e para frente, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para a diagonal da esquerda, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 5, “b”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (5, B)

Teste 89: Movimento para a direita, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando para a direita, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “d”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (4, D)

Teste 90: Movimento para a esquerda, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando para a esquerda, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “b”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (4, B)

Teste 91: Movimento para trás, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o peão está andando para trás, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “b”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (4, B)

Teste 92: Movimento para a diagonal da direita e para trás, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para a diagonal da direita, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “d”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (3, D)

Teste 93: Movimento para a diagonal da esquerda e para trás, do rei em uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para a diagonal da esquerda, o que é válido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “b”).

Valor de retorno: true

Tela: Testando Rei

Movimento válido! Sua posição atual é: (3, B)

Teste 94: Movimento para frente, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em mais de uma casa para frente, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “c”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 95: Movimento para trás, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em uma casa para trás, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “c”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 96: Movimento para a diagonal da direita e para frente, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em mais de uma casa para a diagonal da direita, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “d”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 97: Movimento para a diagonal da esquerda e para frente, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em mais de uma casa para a diagonal da esquerda, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 6, “b”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 98: Movimento para a direita, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando para a direita em mais de uma casa, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “e”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 99: Movimento para a esquerda, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando para a esquerda em mais de uma casa, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 4, “a”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 100: Movimento para trás, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando para trás em mais de uma casa, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 2, “c”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 101: Movimento para a diagonal da direita e para trás, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em mais de uma casa para a diagonal da direita, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “e”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 102: Movimento para a diagonal da esquerda e para trás, do rei em mais de uma casa.

Objetivo do teste: O objetivo do teste é verificar se o rei está andando em mais de uma casa para a diagonal da esquerda, o que é inválido.

Objeto: novoR.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “a”).

Valor de retorno: false

Tela: Testando Rei

Movimento inválido! Sua posição atual é: (4, C)

Teste 103: Movimento de peça violando coordenada linha.

Objetivo do teste: O objetivo do teste é verificar se é possível inserir em uma posição fora do tabuleiro, o que é inválido.

Objeto: novoP, novaT, novoC, novoB, novoR, novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 9, “a”).

Valor de retorno: false

Tela: Testando <nome da peça>

Voce violou as dimensões do tabuleiro!

Movimento inválido! Sua posição atual é: (4, C)

Teste 104: Movimento de peça violando coordenada coluna.

Objetivo do teste: O objetivo do teste é verificar se é possível inserir em uma posição fora do tabuleiro, o que é inválido.

Objeto: novoP, novaT, novoC, novoB, novoR, novaD.

Método testado: checaMovimento(linhaOrigem,colunaOrigemInt, linhaDestino, colunaDestinoInt).

Valores dos parâmetros: checaMovimento(4, ”c”, 3, “i”).

Valor de retorno: false

Tela: Testando <nome da peça>
Voce violou as dimensões do tabuleiro!
Movimento inválido! Sua posição atual é: (4, C)

5. Conclusão

Neste trabalho abordei o Jogo de Xadrez e desenvolvi a primeira fase do projeto, apresentando algumas características e conceitos da programação orientada a objetos.

Foi possível usar diversos conhecimentos obtidos com a disciplina, assim, notei algumas mudanças que poderei fazer para a segunda fase como, por exemplo, o uso de herança na hora de criação dos tipos de peça do jogo.

Além disso, foi possível observar a importância de fazer diversos testes para a correção e verificação da lógica de movimentação das peças.

Este trabalho foi muito importante para meu conhecimento e aprofundamento no desenvolvimento da programação, uma vez que me permitiu aperfeiçoar competências de investigação, organização e criação e acredito que tenha sido possível cumprir com todos os objetivos que foram propostos, ganhando experiência e aprendizagem.