

UNIVERSIDADE FEDERAL DE SÃO CARLOS  
DEPARTAMENTO DE COMPUTAÇÃO DE SOROCABA

**Programação Orientada a Objetos**

**Relatório do Projeto Final de um Jogo de Xadrez**

Matheus Fernando Vieira Pinto

## **1.Introdução**

Um jogo de xadrez é composto de 1 tabuleiro (composto de 64 posições), 32 peças (sendo 16 brancas e 16 pretas) e dois jogadores. O objetivo deste relatório é a descrição de como o jogo de xadrez foi implementado com a utilização dos conceitos e paradigmas da Programação Orientada a Objetos (POO).

## **2.Implementação das Classes**

Ao todo foram implementadas 12 classes, cada uma representando um elemento do jogo de xadrez, são elas as classes Peca, Rei, Rainha, Bispo, Cavalo, Torre, Peao, Posicao, Tabuleiro, Jogador, Jogo e Gerenciador. A seguir serão descritas cada uma dessas classes individualmente, apresentando seus componentes e funcionalidade.

### **2.1 Classe Peca**

Para generalizar os atributos e métodos comuns às peças do jogo de xadrez uma classe Peca que neste trabalho é abstrata foi implementada para agrupar esses dados de modo a utilizar os conceitos de herança proporcionados pela programação orientada a objetos. A classe Peca é composta de 3 atributos, 1 método construtor, 3 métodos assessores, 1 método modificador e 1 método de instância que serão descritos a seguir.

#### **2.1.1 - Atributos**

- cor - atributo do tipo caractere que representa a cor de uma determinada peça.
- capturada - atributo do tipo lógico que determina se uma determinada peça está capturada (verdadeiro) ou não (falso).
- podePular - atributo do tipo lógico que determina se uma determinada peça pode pular(verdadeiro) ou não(falso).

#### **2.1.2 - Construtor**

O método construtor da classe peça recebe como parâmetro um caractere que representa a cor da peça a ser instanciada, ele inicializa os atributos cor, capturada e podePular apenas se o valor fornecido como parâmetro for os caracteres B (branco) ou P (preto), caso isso não aconteça ele gera uma exceção informando o erro ocorrido.

#### **2.1.3 - Métodos Acessores e Modificadores**

- getCor - retorna um caractere que representa a cor de uma determinada peça, para este projeto existem dois possíveis B (peças brancas) ou P (peças pretas).

- `getCapturada` - retorna um valor lógico que indica se uma peça está capturada (verdadeiro) ou não (falso).
- `getPodePular` - retorna um valor lógico que indica se uma determinada peça pode pular (verdadeiro) ou não (falso).
- `setCapturada` - recebe como parâmetro um valor lógico e modifica o valor do atributo capturada.

#### 2.1.4 - Métodos de instância abstratos

- `desenho` - método abstrato que retorna um caractere que representa o desenho de uma determinada peça. Esse método não é implementado nessa classe, mas deve ser implementado nas classes filhas dela.
- `checaMovimento` - método abstrato que recebe como parâmetro a linha de origem (inteiro), coluna origem (caractere), linha destino (inteiro), coluna de destino (caractere) respectivamente e retorna um valor lógico que indica se o movimento da peça é válido (verdadeiro) ou não (falso). Esse método não é implementado nessa classe, mas deve ser implementado nas classes filhas dela.

## 2.2 Classe Rei

De acordo com as regras do jogo de xadrez toda peça Rei pode se mover em qualquer direção (diagonal, vertical e horizontal) uma casa de cada vez desde que não esteja ameaçada (ou fique em xeque). Para um objeto da classe Rei não interessa saber se ele está ameaçado ou não, pois esta é a função de outra classe. A classe Rei implementada neste projeto é uma subclasse da superclasse Peca e portanto possui todos os atributos e métodos da superclasse Peca que foram apresentados na seção 2.1.

### 2.2.1) Construtor

O construtor Rei é do tipo público e recebe um parâmetro chamado `cor` que é do tipo caractere, o construtor inicializa alguns atributos herdados de forma automática e outros de acordo com o parâmetro fornecido pelo usuário da classe. Ele inicializa o atributo `cor` com o valor do parâmetro fornecido, o atributo `capturada` com falso (ou seja, ela está ativa no início do jogo) e o atributo `podePular` com falso (a peça rei não pode pular). Para inicializar esses atributos é necessário chamar o método construtor da classe mãe que foi discutido na seção 2.1.2.

### 2.2.2) Métodos acessores e modificadores:

Os métodos acessores e modificadores da classe rei são os mesmos já discutidos na seção 2.1.3 já que eles são herdados da classe peça.

### 2.2.3) Métodos de Instância:

Os métodos de instância da classe rei são os herdados da superclasse peça (seção 2.1.4), porém eles devem ser implementados de forma polimórfica aqui já que são métodos abstratos na classe mãe.

- `desenho ()` - retorna o formato do rei, ou seja, um caractere podendo ser R (rei branco) ou r (rei preto) de acordo com a cor armazenada no atributo `cor`.
- `checaMovimento` – esse é o método que irá analisar todos os possíveis movimentos de um rei ele recebe como parâmetros as coordenadas da posição de origem (`linhaOrigem`, `colunaOrigem`) e as coordenadas da posição de destino (`linhaDestino`, `colunaDestino`) e retorna um booleano verdadeiro se o movimento foi possível e um booleano falso caso contrário. Para a implementação desse método foi usada como base a regra do jogo de xadrez que determina que um rei pode se mover para qualquer lado (diagonal, horizontal e vertical) uma casa de cada vez. Como exemplo genérico assumo um rei que está na posição de coordenadas (`x`, `y`), logo ele pode se mover para as posições de coordenadas (`x-1`, `y-1`), (`x-1`, `y`), (`x-1`, `y+1`), (`x`, `y-1`), (`x`, `y+1`), (`x+1`, `y-1`), (`x+1`, `y`) e (`x+1`, `y+1`).

## 2.3 Classe Bispo

No jogo de xadrez uma peça bispo pode se mover quantas posições ela quiser na diagonal desde que não existam peças no caminho e que ela não ultrapasse o limite do tabuleiro. A classe bispo implementada neste projeto é uma subclasse da superclasse `Peca` e portanto herdará todos os atributos e métodos da classe `Peca`.

### 2.3.1) Atributos

A classe bispo possui apenas os atributos herdados da classe mãe, assim como apresentado na seção 2.1.1.

### 2.3.2) Construtor

O construtor da classe Bispo recebe 1 parâmetro do tipo caractere que representa a cor da peça, ele inicializa com o auxílio do construtor da classe mãe (seção 2.1.2) os atributos herdados cor com o parâmetro fornecido. Além disso, ele atribui o valor falso ao atributo podePular (a peça bispo não pula).

#### 2.3.3) Métodos acessores e Modificadores

Os métodos gets e sets da classe bispo são os mesmos da classe Peca (seção 2.1.3) já que existe uma relação de herança entre elas.

#### 2.3.4) Métodos de instância

Os métodos de instância são os mesmos da classe Peca, porém eles são sobrepostos polimorficamente já que são abstratos na classe mãe.

- desenho – método que retorna um caractere que representa o formato do bispo B (bispo branco) ou b (bispo preto) de acordo com o valor do atributo cor. Esse método não recebe parâmetros.
- checaMovimento – método responsável por validar o movimento do bispo recebe 4 parâmetros são eles: linhaOrigem (inteiro), colunaOrigem (char), linhaDestino (inteiro) e colunaDestino (char). Possui como valor de retorno um tipo lógico (verdadeiro - movido ou falso - falha). Para a implementação deste método foi utilizada a regra do xadrez (bispo só se move na diagonal de acordo com a sua cor) é uma fórmula matemática (verificar se o módulo da diferença entre a linha de destino e a linha de origem é igual ao módulo da diferença entre a coluna de destino e a coluna de origem).

## 2.4 Classe Torre

No jogo de xadrez uma peça torre pode se mover na horizontal ou vertical quantas posições desejar, mas para que isso ocorra não deve existir peças no seu caminho. A classe Torre implementada neste projeto é uma subclasse da superclasse Peca que já foi apresentada na seção 2.1, e portanto herda todos os atributos e métodos da classe Peca.

#### 2.4.1) Atributos

A classe torre possui apenas os atributos herdados da classe Peca (seção 2.1.1).

#### 2.4.2) Construtor

O construtor da classe Torre recebe 1 parâmetro do tipo caractere que representa a cor da peça, ele inicializa com o auxílio do construtor da superclasse Peça o atributo cor de acordo com o parâmetro fornecido. Além disso, ele atribui o valor falso ao atributo podePular (uma torre não pode pular).

#### 2.4.3) Métodos acessores e Modificadores

Os métodos gets e sets da classe torre são os mesmos da classe Peca (seção 2.1.3) já que existe uma relação de herança entre elas.

#### 2.4.4) Métodos de instância

Os métodos de instância da classe Torre são os mesmos da classe Peca, mas nesse caso eles foram sobrepostos com a utilização da herança e do polimorfismo.

- **desenho** – método que retorna um caractere que representa o formato da torre T (peças brancas) e t (peças pretas) de acordo com o atributo cor herdado da superclasse Peca. Esse método não recebe parâmetros.
- **checaMovimento** – método responsável por validar o movimento da torre recebe 4 parâmetros são eles: linhaOrigem (inteiro), colunaOrigem (char), linhaDestino (inteiro) e colunaDestino (char). Possui como valor de retorno um tipo lógico (verdadeiro - movido ou falso - falha). Para a implementação deste método foi utilizada a regra do xadrez, ou seja a torre só pode se mover para qualquer posição em que a linha de destino seja igual a linha origem (movimento horizontal) ou a coluna de destino seja igual a coluna de origem (movimento vertical).

## 2.5 Classe Cavalo

No jogo de xadrez uma peça do tipo cavalo possui movimento no formato de L e pode pular peças existentes entre o caminho de deslocamento. A classe Cavalo implementada neste projeto é uma subclasse da superclasse Peca e herda todos os atributos e métodos desta superclasse.

### 2.5.1) Atributos

- **cor** – atributo do tipo caractere herdado de Peca que armazena a cor do cavalo (B – branca e P – preta)
- **capturada** – atributo do tipo lógico herdado de Peca que armazena o status do cavalo podendo ele estar em jogo ou não.
- **podePular** - atributo do tipo lógico herdado de Peca que indica se um cavalo pode pular ou não.

### 2.5.2) Construtor

O construtor da classe Cavalo recebe 1 parâmetro do tipo caractere que representa a cor da peça e inicializa o atributo cor com o auxílio do construtor da superclasse Peca. Além disso, ele atribui o valor verdadeiro ao atributo podePular, já que um cavalo pode pular peças durante o seu movimento.

### 2.5.3) Métodos acessores e Modificadores

Os métodos gets e sets da classe Cavalo são herdados da classe Peca.

- **setCapturada** – recebe um parâmetro do tipo lógico e modifica o conteúdo da variável de instância capturada. Esse método não possui retorno.
- **getCapturada** – retorna o conteúdo armazenado no atributo capturada. Esse método não recebe parâmetros.
- **getCor** – retorna o conteúdo do atributo cor. Esse método não recebe parâmetros.

- `getPodePular` - retorna verdadeiro se o cavalo pode pular e falso caso contrário. Esse método não possui parâmetros.

#### 2.5.4) Métodos de instancia

Os métodos de instância da classe Cavalo são herdados da superclasse Peca e sobrepostos com o auxílio do polimorfismo, uma vez que existe uma relação de herança entre elas e esses métodos são abstratos na classe mãe.

- `desenho` – método que retorna um caractere (C - peças brancas ou c - peças pretas) que representa o desenho de um cavalo de acordo com o atributo `cor`. Esse método não recebe parâmetros.
- `checaMovimento` – método responsável por validar o movimento do cavalo recebe 4 parâmetros são eles: `linhaOrigem` (inteiro), `colunaOrigem` (char), `linhaDestino` (inteiro) e `colunaDestino` (char). Possui como valor de retorno um tipo lógico (verdadeiro - movido ou falso - falha). Para a implementação deste método foi utilizada a regra do xadrez (o cavalo se move em L e pode pular peças). Como exemplo pegue um cavalo genérico na posição (x, y) as posições de destino que validam o movimento do cavalo são (x-2, y-1), (x-2, y+1), (x-1, y-2), (x-1, y+2), (x+1, y-2), (x+1, y+2), (x+2, y-1) e (x+2, y+1).

## 2.6 Classe Rainha

No jogo de xadrez uma rainha pode se movimentar na diagonal e na vertical na quantidade de casas que ela desejar, porém não deve existir peças no seu caminho. A classe Rainha implementada neste projeto é uma subclasse da superclasse Peca, logo herda todos os atributos e métodos da classe Peca.

### 2.6.1) Atributos

A classe Rainha possui apenas os atributos herdados da Peca (herança simples).

- `cor` – atributo do tipo caractere que armazena a cor da rainha (B – branca e P – preta).
- `capturada` – atributo do tipo lógico que armazena o status da rainha podendo ela estar em jogo ou não.
- `podePular` - atributo do tipo lógico que indica se uma rainha pode pular ou não.

### 2.6.2) Construtor

O construtor da classe Rainha recebe 1 parâmetro do tipo caractere que representa a cor da peça e inicializa o atributo `cor` com o auxílio do construtor da superclasse Peca. Além disso, ele atribui o valor falso ao atributo `podePular`, já que uma rainha não pode pular peças durante o seu movimento.

### 2.6.3) Métodos acessores e modificadores

Os métodos gets e sets da classe Rainha são herdados da classe Peca.

- `setCapturada` – recebe um parâmetro do tipo lógico e modifica o conteúdo da variável de instância `capturada`. Esse método não possui retorno.

- `getCapturada` – retorna o conteúdo armazenado no atributo `capturada`. Esse método não recebe parâmetros.
- `getCor` – retorna o conteúdo do atributo `cor`. Esse método não recebe parâmetros.
- `getPodePular` - retorna verdadeiro se o rainha pode pular e falso caso contrário. Esse método não possui parâmetros.

#### 2.6.4) Métodos de instancia

Os métodos de instância da classe `Rainha` são herdados da superclasse `Peca` e sobrepostos com o auxílio do polimorfismo, uma vez que existe uma relação de herança entre elas e esses métodos são abstratos na classe mãe.

- `desenho` – método que retorna um caractere (D - peças brancas ou d - peças pretas) que representa o desenho de uma rainha de acordo com o atributo `cor`. Esse método não recebe parâmetros.
- `checaMovimento` – método responsável por validar o movimento da rainha recebe 4 parâmetros são eles: `linhaOrigem` (inteiro), `colunaOrigem` (char), `linhaDestino` (inteiro) e `colunaDestino` (char). Possui como valor de retorno um tipo lógico (verdadeiro - movido ou falso - falha). Para a implementação deste método foi utilizada a regra do xadrez, onde o movimento da rainha é a combinação dos movimentos de um bispo e de uma torre.

### 2.7 Classe Peao

No jogo de xadrez um peão pode se movimentar somente para frente na vertical ou diagonal uma casa de cada vez. Ele pode se movimentar duas casas na vertical apenas quando realiza seu primeiro movimento. O movimento na diagonal só é permitido quando existe uma peça inimiga a ser capturada na posição de destino, já o movimento na vertical é permitido apenas quando não existe peça na posição de destino. A classe `Peao` implementada neste projeto é uma subclasse da superclasse `Peca`, ela herda todos os métodos e atributos da classe `Peca`. Diferente das outras classes filhas de `Peca`, a classe `Peao` possui um atributo específico dela.

#### 2.7.1) Atributos

A classe `Peao` possui todos os atributos herdados da classe `Peca` e o atributo `primeiroMovimento`.

- `cor` – atributo do tipo caractere que armazena a cor do peão (B – branca e P – preta)
- `capturada` – atributo do tipo lógico que armazena o status do peão, podendo ele estar em jogo ou não.
- `primeiroMovimento` - atributo do tipo lógico que indica se é o primeiro movimento de um peão.



### 2.7.2) Construtor

O construtor da classe Peao recebe 1 parâmetro do tipo caractere que representa a cor da peça, ele inicializa com o auxílio do construtor da superclasse Peca os atributos cor com o parâmetro fornecido, o atributo podePular como falso e o atributo capturada com falso. Além disso, ele atribui o valor verdadeiro ao atributo primeiroMovimento.

### 2.7.3) Métodos acessores e Modificadores

Os métodos gets e sets da classe peão são os mesmos da superclasse Peca mais o get relacionado ao atributo primeiroMovimento.

- setCapturada – recebe um parâmetro do tipo lógico e modifica o conteúdo da variável de instância capturada. Esse método não possui retorno.
- getCapturada – retorna o conteúdo armazenado no atributo capturada. Esse método não recebe parâmetros.
- getCor – retorna o conteúdo do atributo cor. Esse método não recebe parâmetros.
- getPodePular - retorna sempre falso uma vez que um peão nunca pode pular em um jogo de xadrez. Esse método não possui parâmetros.
- getPrimeiroMovimento - retorna verdadeiro se é o primeiro movimento do peão e falso caso contrário. Esse método não possui parâmetros.

### 2.7.4) Métodos de instancia

Os métodos de instância da classe Peao são herdados da superclasse Peca e sobrepostos com o auxílio do polimorfismo, uma vez que existe uma relação de herança entre elas e esses métodos são abstratos na classe mãe. Além disso, essa classe possui um método específico seu que altera o primeiroMovimento do peão de forma definitiva.

- desenho – método que retorna um caractere (P - peões brancos ou p - peões pretos) que representa o desenho do peão de acordo com a sua cor. Esse método não recebe parâmetros.
- checaMovimento – método responsável por validar o movimento do peão recebe 4 parâmetros são eles: linhaOrigem (inteiro), colunaOrigem (char), linhaDestino (inteiro) e colunaDestino (char). Possui como valor de retorno um tipo logico (verdadeiro - movido ou falso - falha). Para a implementação deste método foi utilizada a regra do xadrez, que diz que um peão só pode mover uma casa de cada vez (com exceção da primeira jogada) para frente na vertical ou nas diagonais (para comer uma peça inimiga vizinha). Além disso, a cor do peão determina o sentido do seu movimento (nesse projeto foi considerado que um peão preto se desloca de cima para baixo enquanto que um peão branco se move de baixo para cima). Como exemplo assumo um peão branco genérico localizado na posição (x, y) as possíveis posições de destino são: (x+1, y), (x+1, y-1), (x+1, y+1), (x+2, y). Agora assumo um peão preto genérico na mesma posição as possíveis posições de destino são: (x-1, y), (x-1, y-1), (x-1, y+1), (x-2, y).

- movido - método que não possui parâmetros nem valor de retorno, apenas modifica o valor do atributo primeiroMovimento para false.

## 2.8 Classe Posicao

A classe Posicao implementada possui 5 atributos, 1 construtor, 5 métodos acessores e 2 métodos modificadores.

### 2.8.1) Atributos

- cor – atributo do tipo caractere que armazena a cor da posição (B – branca ou P – preta).
- linha – atributo do tipo inteiro que armazena a linha em que a posição está localizada.
- coluna – atributo do tipo caractere que armazena a coluna em que a posição está localizada.
- estaOcupada – atributo do tipo lógico que indica se uma posição está ocupada ou não.
- peca - uma referência para um objeto do tipo Peca, esse atributo armazena uma determinada peça na posição solicitada.

### 2.8.2) Construtor

O construtor da classe Posicao recebe 4 parâmetros (cor – caractere, linha – inteiro e coluna – caractere, peca - referência para um objeto do tipo peça) e inicializa os atributos cor, linha, coluna e peça de acordo com o valor desses parâmetros. Além disso, ele também inicializa de forma automática o conteúdo da variável de instância estaOcupada para falso. O método construtor também verifica se os parâmetros fornecidos são válidos, caso eles não sejam, ele dispara uma exceção indicando o erro.

### 2.8.3) Métodos Acessores e Modificadores

- getCor – esse método não recebe parâmetros, apenas retorna o conteúdo do atributo cor que é um caractere..
- getLinha – esse método não recebe parâmetros, apenas retorna o conteúdo do atributo linha que é um inteiro.
- getColuna – esse método não recebe parâmetros, apenas retorna o conteúdo do atributo coluna que é um caractere.
- getEstaOcupada – retorna um valor lógico que indica se a posição está ocupada ou não. Não recebe parâmetros.
- setEstaOcupada – recebe um valor do tipo lógico como parâmetro e o insere no atributo estaOcupada. Esse método não possui retorno.
- getPeca - retorna uma referência a um objeto do tipo Peca que está armazenado na posição. Esse método não possui parâmetros.

- setPeca - recebe como parâmetro uma referência para um objeto do tipo Peca e atribui o valor dessa referência ao atributo peça. Este método não possui valor de retorno.

## 2.9 Classe Tabuleiro

A classe tabuleiro implementada possui 1 atributo, 1 construtor e 10 métodos de instância.

### 2.9.1) Atributos

- tabuleiro – essa variável é do tipo ArrayList<Posicao> da biblioteca java.util.ArrayList que nada mais é que uma lista de objetos da classe Posicao. Ao todo serão inseridos 64 posições nessa lista.

### 2.9.2) Construtor

O construtor da classe Tabuleiro não recebe parâmetros, seu principal objetivo é inserir as 64 posições que representam um tabuleiro de xadrez dentro do atributo tabuleiro. A cada posição inserida no tabuleiro é atribuída a ela uma cor (B ou P), linha (1 até 8) e coluna(a até h).

### 2.9.3) Métodos de Instância

- buscaIndicePosicao – método que retorna o índice (0 a 63) da posição procurada no atributo tabuleiro (lista de objetos do tipo Posicao) caso ela exista. Recebe como parâmetros a linha (inteiro) e a coluna (caractere) da posição procurada. Retorna o valor do índice caso a posição seja encontrada ou o valor -1 caso a posição não seja encontrada. Além disso, ele gera uma exceção caso a linha ou coluna fornecida seja inválida.
- imprimeTabuleiro - método utilizado para imprimir o tabuleiro no formato de uma matriz na qual as linhas são representadas por números de 1 até 8 e as colunas representadas por letras de a até h. Cada elemento da matriz é representado pelo desenho da peça que ocupa a posição ou pelo caractere ‘\*’ caso a posição esteja vazia. Esse método não possui valor de retorno.
- movePeca - método que realiza o movimento de uma peça dentro do tabuleiro, ele move a peça para a posição de destino e atribui o valor null a posição de origem. Esse método recebe como parâmetro a linha origem (int), coluna origem (char), linha destino (int), coluna destino (int) e uma referência para um objeto do tipo Peca. O retorno deste método é a referência à peça que ocupava a posição de destino (ela pode ser null).
- fazMovimento - método que realiza o movimento de uma determinada peça considerando todas as possíveis falhas (se existem peças no caminho, o movimento do peão, se o movimento da peça é válido, etc) durante um movimento de uma peça. Recebe como parâmetro a linha origem (int), coluna origem (char), linha destino (int), coluna destino (int) e uma referência para um objeto do tipo Peca. O retorno deste método é um valor lógico que indica se o movimento é valido (true) ou não (false).

- desfazMovimento - método que realiza o movimento inverso ao realizado pelo método movePeca, além de modificar o status da peça removida caso ela não seja null para verdadeiro. Possui como parâmetros a linha origem (int), coluna origem (char), linha destino (int), coluna destino (int), uma referência para a peça movida e uma referência para a peça removida. Não possui valor de retorno.
- haPecasNoCaminho - método que verifica se existem peças no caminho durante o movimento de uma determinada peça dentro do tabuleiro. Esse método recebe como parâmetro as coordenadas (linha e coluna) da posição de origem (linha e coluna) e as coordenadas da posição de destino respectivamente. Esse método retorna um valor lógico indicando se existem peças no caminho (verdadeiro) ou não (falso).
- inserePeca - método para inserir uma determinada peça em uma posição específica do tabuleiro. Recebe como parâmetro a linha (int) e coluna (char) da posição onde a peça será inserida e a referência à peça. Não possui valor de retorno.
- quemEstaPosicao - método que retorna a peça que ocupa uma determinada posição do tabuleiro. Recebe como parâmetro a linha (int) e coluna (char) da posição do tabuleiro de onde se deseja obter a peça.
- linhaPeca - método que retorna em qual linha do tabuleiro está localizada uma determinada peça. Recebe como parâmetro uma referência para uma peça e retorna um inteiro indicando a linha onde essa peça está localizada.
- colunaPeca - método que retorna em qual coluna do tabuleiro está localizada uma determinada peça. Recebe como parâmetro uma referência para uma peça e retorna um caractere indicando a linha onde essa peça está localizada.

## 2.10 Classe Jogador

A classe jogadora é composta de 3 atributos, 1 construtor, 2 métodos acessores, 2 métodos modificadores e 3 métodos de instância.

### 2.10.1) Atributos

- nome – atributo privado do tipo String que armazena o nome do jogador.
- pecas - atributo do tipo composto que armazena todas as peças de um determinado jogador.
- pecasRestante – atributo privado do tipo inteiro que armazena o número de peças ainda restantes.

### 2.10.2) Construtor

O construtor da classe Jogador recebe como parâmetro uma String que será utilizada para inicializar o atributo nome, além disso ele também inicializa o atributo pecasRestantes para 16.

### 2.10.3) Métodos Acessores e Modificadores

- getNome – retorna o conteúdo do atributo nome (String), não possui parâmetros.

- `getPecasRestantes` – retorna o conteúdo do atributo `pecasRestantes` (inteiro), não possui parâmetros.
- `setPecasRestantes` – recebe um parâmetro do tipo inteiro que será utilizado para modificar o conteúdo da variável `pecasRestantes`. Não possui valor de retorno.
- `setPecas` - recebe uma referência para um conjunto de peças e inicializa o atributo `pecas` com ela. Não possui valor de retorno

#### 2.10.4) Métodos de Instância

- `imprimePecasCapturadas` - método que imprime todas as peças capturadas de um jogador.
- `retornaPecasAtivas` - método que obtém todas as peças ativas de um jogador e retorna uma referência para este conjunto (No caso deste projeto um `ArrayList<Peca>`). Não recebe parâmetros.
- `procuraRei` - método que busca um rei de um determinado jogador. Retorna a referência para a peça que representa o rei deste jogador, não recebe parâmetros.

### 2.11 Classe Jogo

A classe `jogo` é composta por 6 atributos, 2 construtores, 2 métodos acessores, 2 métodos modificadores e 11 métodos de instância.

#### 2.11.1) Atributos

- `jogador` – atributo composto do tipo `Jogador` (possui 2 jogadores) .
- `tabuleiro` – atributo privado do tipo `tabuleiro`.
- `vezDeQuem` – atributo do tipo inteiro que indica de qual jogador é a vez (jogador 1 – 0 e jogador 2 – 1).
- `estadoJogo` – variável do tipo inteiro que indica em qual estado o jogo se encontra. -1(início), 0(xeque) e 1(xeque-mate)
- `pecasBrancas` - atributo do tipo `ArrayList<Peca>` que armazena todas as 16 peças brancas do jogo de xadrez.
- `pecasPretas` - atributo do tipo `ArrayList<Peca>` que armazena todas as 16 peças pretas do jogo de xadrez.

#### 2.11.2) Construtor

O construtor 1 da classe `Jogo` recebe como parâmetro dois objetos do tipo `jogador` e 1 inteiro que serão utilizados para inicializar os atributos `jogador[0]`, `jogador[1]` e `vezDeQuem`. Ele também cria e distribui as peças entre os jogadores (jogador 1 com peças brancas e jogador 2 com peças pretas) e as organiza no tabuleiro (As peças brancas na parte inferior do tabuleiro e as pretas na parte superior).

O construtor 2 da classe `tabuleiro` não recebe parâmetros e retoma um jogo a partir de um arquivo pré-estabelecido.

### 2.11.3) Métodos Acessores e Modificadores

- `getVezDeQuem` – retorna o conteúdo do atributo `vezDeQuem` (inteiro), não possui parâmetros.
- `getEstadoJogo` – retorna o conteúdo do atributo `estadoJogo` (inteiro), não possui parâmetros.
- `setVezDeQuem` – recebe um parâmetro do tipo inteiro que será utilizado para modificar o conteúdo da variável `vezDeQuem`. Não possui valor de retorno.
- `setEstadoJogo` – recebe como parâmetro um inteiro que será utilizado para modificar o conteúdo do atributo `estadoJogo` (depende de quem é o jogador atual). Não possui valor de retorno.

### 2.11.4) Métodos de instância.

- `inicializaPecas` - método que cria todas as peças do jogo e as armazena no atributos `pecasBrancas` e `pecasPretas` de acordo com a cor de cada uma. Esse método não possui parâmetros nem retorno.
- `distribuiPecas` - método que distribui as peças entre os jogadores, o jogador de índice 0 recebe as peças brancas e o jogador de índice 1 recebe as peças pretas. Esse método não possui parâmetros nem retorno.
- `organizaTabuleiro` - método que dispõe as peças no tabuleiro de acordo com a organização padrão do jogo de xadrez. neste projeto as peças brancas ficam na parte inferior do tabuleiro e as pretas na parte superior. Esse método não possui parâmetros nem retorno.
- `escolheOrigem` - método que obtém uma determinada peça a ser movimentada. Este método verifica se a peça da posição escolhida pelo jogador pertence a ele ou não e se ele não escolheu uma posição vazia (nesses casos uma exceção é gerada). Recebe como parâmetro a linha (int) e coluna (char) da posição do tabuleiro onde está localizada a peça a ser movimentada retorna a referência para a peça caso ela pertença ao jogador.
- `fazJogada` - método que verifica se a jogada do jogador é válida ou não (nesse caso gera uma exceção indicando movimento invalido). Recebe como parâmetro as coordenadas de origem, as coordenadas de destino e a peça a ser manipulada. Não possui valor de retorno.
- `testaXeque` - método que verifica se o rei do jogador atual está em xeque, ou seja, uma das peças do jogador oponente consegue fazer um movimento até o rei do jogador atual. Retorna verdadeiro caso o rei esteja em xeque e falso caso contrário. Esse método não recebe parâmetros.
- `testaXequeMate` - método que verifica se o jogo está em xeque mate, ou seja, nenhuma das peças do jogador alvo pode tirar o rei de um xeque. Retorna verdadeiro caso o rei esteja em xeque mate e falso caso contrário.
- `gravarArquivo` - método que armazena os atributos do objeto jogo para recuperá-lo posteriormente. Recebe como parâmetro o nome do arquivo (String) no qual o jogo será armazenado. Não possui valor de retorno.
- `lerArquivo` - método para ler os atributos de um jogo gravado em um arquivo e inicializar os atributos do jogo. Recebe como parâmetro uma string que indica o nome do arquivo de onde os dados serão lidos. Não possui valor de retorno.
- `limpaTela` - método para limpar a tela constantemente.

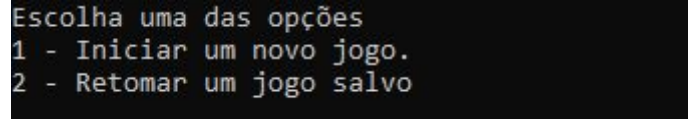
- rodada - método utilizado como interface principal do jogo, é nesse método que a interação entre os jogadores acontece e também é onde as exceções são tratadas.

## 2.12 Classe Gerenciador

É nesta classe que o método main está localizado e é nela que as informações referentes aos jogadores são obtidas. É nesta classe que o jogo é inicializado ou retomado de acordo com a escolha do usuário.

### 3) Tutorial de como executar e jogar o jogo

1. Antes de executar o jogo é necessário compilar todas as classes .java presentes no pacote.
2. Após a compilação das classes .java é necessário executar o bytecode obtido a partir da compilação da classe Gerenciador.
3. Quando executado um menu com duas opções irá aparecer na tela. O usuário deve escolher se deseja iniciar um novo jogo (1) ou retomar um jogo passado (2).

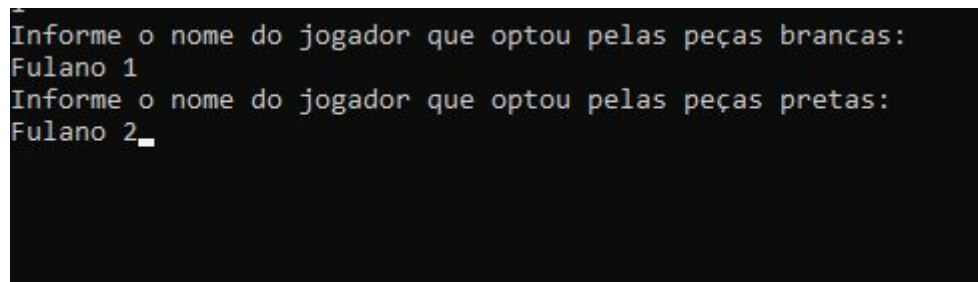


```
Escolha uma das opções
1 - Iniciar um novo jogo.
2 - Retomar um jogo salvo
```

Figura 1: Menu inicial do jogo de xadrez

Fonte: Elaboração própria.

4. Se o usuário optar por iniciar um novo jogo (1) o console irá solicitar o nome dos jogadores da partida.



```
Informe o nome do jogador que optou pelas peças brancas:
Fulano 1
Informe o nome do jogador que optou pelas peças pretas:
Fulano 2_
```

Figura 2: Console solicita o nome dos jogadores.

Fonte: Elaboração própria.

5. Após informar os nomes dos jogadores o sistema irá iniciar o jogo e solicitar do jogador das peças brancas a linha e coluna de origem separadas por um espaço e em seguida a linha e coluna de destino separadas por um espaço, depois de

realizar o movimento a rodada passa a ser do outro jogador e esse processo se repete até que o jogo esteja em xeque-mate.

```

      a b c d e f g h
8 |t|c|b|d|r|b|c|t|
7 |p|p|p|p|p|p|p|p|
6 |*|*|*|*|*|*|*|*|
5 |*|*|*|*|*|*|*|*|
4 |*|*|*|*|*|*|*|*|
3 |*|*|*|*|*|*|*|*|
2 |P|P|P|P|P|P|P|P|
1 |T|C|B|D|R|B|C|T|

Pecas Brancas Capturadas:

Pecas Pretas Capturadas:

Rodada do jogador Fulano 1 movimento as peças brancas
Coordenadas de origem:
2 d
Coordenadas de destino:
4 d

```

Figura 3: Interface principal do jogo

Fonte: Elaboração própria.

6. Ao fim de cada rodada o jogador pode optar por interromper o jogo e para isso o sistema fornece a opção de encerrar o jogo. Caso o jogador opte por encerrar o jogo este é armazenado em um arquivo no qual o nome é fornecido pelo próprio usuário, para poder ser retomado posteriormente.

```

      a b c d e f g h
8 |t|c|b|d|r|b|c|t|
7 |p|p|p|p|p|p|p|p|
6 |*|*|*|*|*|*|*|*|
5 |*|*|*|*|*|*|*|*|
4 |*|*|*|*|*|*|*|*|
3 |P|*|*|*|*|*|*|*|
2 |*|P|P|P|P|P|P|P|
1 |T|C|B|D|R|B|C|T|

Para encerrar o jogo digite 1, mas se deseja continuar digite 0

```

Figura 4: O jogo obtém do terminal a vontade do jogador em continuar ou não o jogo.

Fonte: Elaboração própria.



7. Caso o jogador escolha a opção de retomar um jogo (2) a partida será recuperada com base no arquivo fornecido pelo usuário e o jogo começará a partir da etapa 5.
8. Caso os dados fornecidos para o sistemas sejam inválidos, mensagens informando o erro irão aparecer na tela e o sistema solicitará novamente as entradas do usuário.

#### **4) Tratamento de exceções durante a implementação**

Em algumas classes houve a necessidade de gerar exceções para serem tratadas posteriormente de maneira a evitar que os parâmetros fornecidos fossem inválidos. As exceções tratáveis foram geradas a partir da classe `JogoExcecoes` implementada para realização deste projeto, as demais exceções foram geradas a partir das classe `RuntimeException` e `Exception`. Durante a manipulação de arquivos o tratamento de exceções foi obrigatório. A seguir serão discutidas cada exceção.

1° - Classe `Peca`: No método construtor da classe `Peca` uma exceção do tipo `Exception` é gerada caso o caractere fornecido como parâmetro seja diferente de P ou B adotados como convenção para as cores das peças nesse jogo. Uma mensagem como o conteúdo “Cor Inválida” é gerada a fim de indicar a exceção ocorrida.

2° - Classe `Posicao`: O método construtor da classe `Posicao` gera exceções do tipo `Exception` caso o caractere fornecido como parâmetro para a cor seja diferente de P ou B, caso a linha esteja fora do intervalo [1, 8] e caso a coluna esteja fora do intervalo [a,b].

3° - Classe `Tabuleiro`: O método de instância `buscaIndicePosicao` gera uma exceção do tipo `JogoExcecoes` caso as linha fornecida como parâmetro esteja fora do intervalo [1,8] ou a coluna fornecida como parâmetro esteja fora do intervalo [a, h]. O métodos de instância `linhaPeca` e `colunaPeca` geram uma exceção do tipo `Exception` caso a referência para uma peça fornecida como parâmetro seja null.

4° - Classe `Jogador`: O método de instância `retornaPecasAtivas` gera uma exceção do tipo `Exception` caso o jogador não possua mais peças ativas. O método de instância `procuraRei` gera uma exceção do tipo `Exception` caso o jogador não possua uma peça rei.

5° - Classe `Jogo`: Os método `escolheOrigem` pode gerar duas exceções do tipo `JogoExcecoes` a primeira ocorre caso as coordenadas de uma posição vazia no tabuleiro sejam fornecidas como parâmetro e a segunda ocorre quando as coordenadas de uma posição ocupada por uma peça adversária são fornecidas como parâmetro. O método `fazJogada` gera uma exceção do tipo `JogoExcecoes` caso o movimento realizado com base nos parâmetros fornecidos seja inválido.

Todos os outros métodos que fazem uso dos métodos listados anteriormente devem tratar ou repassar as exceções geradas por ele. Na classe `jogo`, o método `rodada` faz o tratamento das exceções a fim de garantir que o jogo não seja interrompido por causa

delas e desse modo o usuário é informado sobre o erro ocorrido e o sistema solicita uma nova leitura do console após o usuário pressionar a tecla enter .

Exemplos de exceções tratadas durante a execução do arquivo.

```

  a b c d e f g h
8 |*|*|b|d|r|b|c|t|
7 |p|*|*|B|p|p|*|*|
6 |*|*|*|*|*|*|p|p|
5 |*|*|p|*|*|*|B|*|
4 |*|*|*|p|*|*|*|*|
3 |*|*|*|*|*|C|*|*|
2 |t|P|P|*|*|P|P|P|
1 |T|C|*|R|T|*|*|*|

Check!
Pecas Brancas Capturadas: |D||P|

Pecas Pretas Capturadas: |p||p||p||c|

Rodada do jogador Fernando movimente as peças pretas
Coordenadas de origem:
2 p
Coluna inválida
```

figura 5

```

  a b c d e f g h
8 |*|*|b|d|r|b|c|t|
7 |p|*|*|B|p|p|*|*|
6 |*|*|*|*|*|*|p|p|
5 |*|*|p|*|*|*|B|*|
4 |*|*|*|p|*|*|*|*|
3 |*|*|*|*|*|C|*|*|
2 |t|P|P|*|*|P|P|P|
1 |T|C|*|R|T|*|*|*|

Check!
Pecas Brancas Capturadas: |D||P|

Pecas Pretas Capturadas: |p||p||p||c|

Rodada do jogador Fernando movimente as peças pretas
Coordenadas de origem:
2 a
Coordenadas de destino:
6 a
Você não se pode colocar em xeque
```

figura 6

```

  a b c d e f g h
8 |*|*|b|d|r|b|c|t|
7 |p|*|*|B|p|p|*|*|
6 |*|*|*|*|*|*|p|p|
5 |*|*|p|*|*|*|B|*|
4 |*|*|*|p|*|*|*|*|
3 |*|*|*|*|*|C|*|*|
2 |t|P|P|*|*|P|P|P|
1 |T|C|*|R|T|*|*|*|

Check!
Pecas Brancas Capturadas: |D||P|

Pecas Pretas Capturadas: |p||p||p||c|

Rodada do jogador Fernando movimente as peças pretas
Coordenadas de origem:
1 a
Essa peça não é sua
```

figura 7

Fonte: Elaboração Própria.

## 5) Diagrama de Classes Versão Final

A figura abaixo representa o diagrama de classe final.

