

Objetivo: O objetivo do nosso projeto é consiste em construir um jogo de xadrez em java, usando conceitos de Programação Orientada a Objetos.

Resumo das classes:

Peca:

Classe abstrata que servirá como classe “Mãe” para cada específica do xadrez.

```
private final String cor;  
private final int valor;  
private Coordenada posicao;  
private final String tag;
```

Tag: Qual peça é (bispo, cavalo, etc)

Valor: Qual o valor da peça para ser contabilizado em uma futura aplicação de placar.

Principais funções:

- getters/setters;
- getPossiveisMovimentos
 - Devolver uma array do tipo Coordenada
 - Abstrata

Todas as peças do xadrez herdam dessa classe “Peca”.

Bispo:

Na função construtora

```
public Bispo(String cor, Coordenada posicao){  
    super(cor, posicao, valor:3, tag:"bispo");  
}
```

Atribuímos de forma padronizada as informações de valor de tag.

Principais funções:

- getPossiveisMovimentos

O bispo anda apenas nas diagonais, a função analisa se há casas vazias ou peças inimigas para serem comidas. A verificação é feita a partir da posição atual do bispo, percorrendo quatro direções na diagonal.

Torre:

Na função construtora

```
public Torre(String cor, Coordenada posicao){  
    super(cor, posicao, valor:5, tag:"torre");  
}
```

Atribuímos de forma padronizada as informações de valor de tag.

Principais funções:

- getPossiveisMovimentos

A torre anda apenas em linha reta, seja na vertical ou horizontal, a função analisa se há casas vazias ou peças inimigas para serem comidas em seu caminho. A verificação é feita a partir da posição atual da torre, percorrendo quatro direções(esquerda, direita, cima, baixo).

Cavalo:

Possui uma lista de movimentos fixos que pode se locomover, coordenadas que serão adicionadas à posição atual para retornar as posições candidatas para se locomover. A partir desta lista de posições candidatas realiza-se a verificação de movimentos possíveis.

```
public Cavalo(String cor, Coordenada posicao){
    super(cor, posicao, valor:3, tag:"cavalo");
}

private final static Coordenada[] MOVIMENTO_PADRAO = {new Coordenada(-2, -1), new Coordenada(-1, -2), new Coordenada(1, -2), new Coordenada(2, -1)}
```

Como descrito anteriormente, possui a lista do tipo Coordenada, com movimentos padrão.

Principais funções:

- getPossiveisMovimentos

Percorre a lista MOVIMENTO_PADRAO, para analisar se há casas vazias ou peças inimigas para serem comidas. Retorna a lista de coordenadas válidas.

Rainha:

Função construtora

```
public Rainha(String cor, Coordenada posicao){
    super(cor, posicao, valor:9, tag:"rainha");
}
```

Principais funções:

- getPossiveisMovimentos

Seu movimento é basicamente a junção do movimento da Torre com o do Bispo. Nessa função verifica-se, as linhas verticais e horizontais, assim como as diagonais. Retorna a lista de coordenadas válidas.

Casa:

Função construtora e atributos

```

private String cor;
private Peca peca;
private JButton botao;
private final Coordenada coord;

public Casa(String cor, int x, int y){
    this.cor = cor;
    this.coord = new Coordenada(x, y);
}

```

Por padrão, não atribuímos nenhuma peça a objetos do tipo Casa. Como Casa não se move ao longo do jogo, definimos sua Coordenada como constante.

Principais funções:

- getters/setters
- boolean estaOcupado()

Verifica se há alguma peça na casa e retorna true caso possua e false caso contrário.

Tabuleiro:

Classe responsável por iniciar todo o tabuleiro. Gerar o tabuleiro com as casas e posicionar as peças em sua posição inicial padrão. Adiciona todos os botões necessários com seus devidos ícones.

Principais funções:

Validacao:

Classe com métodos estáticos que possuem a finalidade de validar diferentes situações

```

public class Validacao {

    public static boolean coordenadaValida(Coordenada coord){
        if(0 > coord.x() || coord.x() > 7)
            return false;

        if(0 > coord.y() || coord.y() > 7)
            return false;

        return true;
    }
}

```

Principais funções:

- getPossiveisMovimentos
 - Devolve se a coordenada recebida está dentro do tabuleiro

- Retorna um valor booleano

Atualmente, a classe conta apenas com o método *coordenadaValida*, porém, ela foi criada para que mais métodos de validação estáticos pudessem ser implementados na mesma.

Coordenada:

Classe responsável por armazenar os valores x e y das coordenadas utilizadas.

```
public class Coordenada{  
    private int x, y;  
  
    public Coordenada(int x, int y){  
        this.x = x;  
        this.y = y;  
    }  
}
```