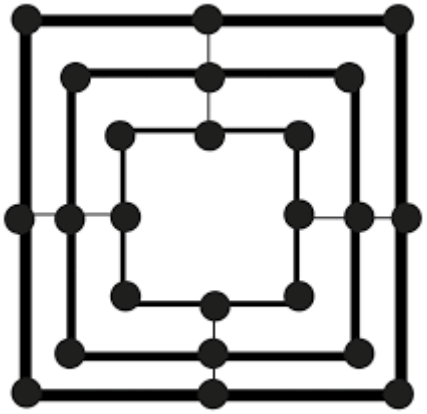
TÉCNICAS DE CONSTRUÇÃO DE PROGRAMA

JOGO TRILHA



GRUPO 7
ERIC D ARRIAGA
JOÃO GABRIEL EBERHARDT PEREIRA
JOÃO VITOR BAGGIO

Classes:

Tabuleiro: Representa o tabuleiro do jogo, e estende Observable para notificar mudanças. Possui as seguintes variáveis e métodos principais:

Atributos:

Casas: Conjunto de 24 casas que representam as intersecções do tabuleiro.

Moinhos: Conjunto de 16 moinhos possíveis de serem formados no tabuleiro.

Jogadores: Array com os dois jogadores do jogo.

Métodos:

colocarPeca, **tirarPeca**, **moverPeca**: Manipulam peças no tabuleiro, como colocar, remover, e mover.

checarMoinho: Verifica se uma trinca foi formada (moinho/trilha).

checarFimjogo: Testa condição final de jogo.

Reiniciar: Reinicia o jogo.

Vencedor: Retorna vencedor no caso de fim de jogo.

Casa: Representa um ponto no tabuleiro onde as peças podem ser colocadas.

Atributos:

x, y: Coordenadas do nó.

corOcupada: Propriedade do nó (WHITE, BLACK ou EMPTY).

Vizinhos: Lista de nós vizinhos para conexão entre nós.

Métodos:

addVizinho: Atualiza lista de vizinhos.

desocupaCasa: Casa passa a ser empty.

ocupaCasa: Casa passa a pertencer a algum jogador.

removeVizinho: Atualiza lista de vizinhos.

Jogador: Representa um jogador no jogo.

Atributos:

seuTurno: Indica se é a vez do jogador.

pecasDisponiveis,**pecasNoTabuleiro:** Quantidade de peças para colocar e peças no tabuleiro.

suaCor: Cor do jogador.

Métodos:

decPecasDisponiveis,**decPecasPerdidas**,**IncPecasPerdidas**,**IncPecasDisponiveis**: Metodos para controlar quantidades de peças dos jogadores.

Moinho: Agrupa três nós para verificar trincas.

Atributos:

casas: Array com as 3 casas que formam o moinho.

Métodos:

temConjunto: Verifica se um nó pertence ao conjunto.

getJogadorMoinho: Retorna o jogador se todos os nós do conjunto são do mesmo jogador.

VisaoDoJogo: Extende Observable e representa a interface gráfica do jogo.

Atributos:

frame: A janela principal do jogo.

Height,Width: Onde o tabuleiro e as peças são desenhados.

Métodos:

+ MyView(observer: Observer)
- criaJanela(): void
- criaComponentes(): void
+ addMensagem(mensagem: String): void
+ mostraMensagem(mensagem: String): void
+ refresh(data: Map<String, Object>

Mensagem: Define mensagens que aparecem na interface de jogo.

Atributos:

Conteudo: String com a mensagem a ser disposta.

tipo: Pode ser, INFO, AVISO ou ERRO.

Métodos:

mensagem: O método mensagem serve para construir a mensagem com o conteúdo dado e o tipo da mensagem.

exibirMsg : Apenas exibe na tela

Controlador: Implementa Observer e coordena as interações entre View e Model.

Atributos:

turno: Indica a vez de qual jogador.

estadoAtual, estadoAnterior: Estado atual e anterior do jogo.

oTabuleiro: Tabuleiro do jogo.

Métodos:

checaFimDeJogo,**checaTirarPeca**,**checaColocarPeca**: Verificam estados do jogo como perda, fase de salto e movimento.

mudaTurno: Alterna a vez entre os jogadores.

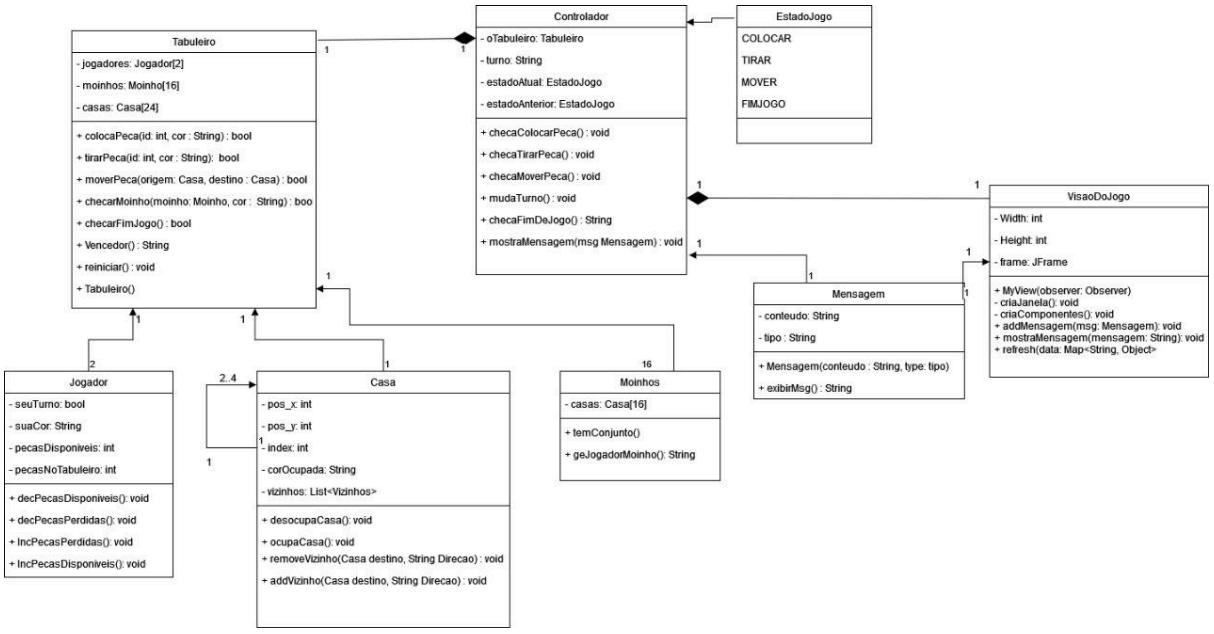
mostraMensagem: Mensagens para guiar o jogo.

Fluxo Geral:

O Controlador recebe as ações do usuário na interface e manipula o Tabuleiro, que notifica a View para atualizar o estado gráfico do jogo. A estrutura MVC (MODEL-VIEW-CONTROLLER)permite separar a lógica do jogo, a interface gráfica e o controle das ações, facilitando a manutenção e a expansão da aplicação.

Observação: Com a finalidade de um relatório mais compacto, não estão incluídos getters e setters inerentes a algumas das classes acima definidas.

DIAGRAMA:



REQUISITOS:

Requisitos Funcionais
RF-1: O sistema deve permitir que dois jogadores joguem no mesmo computador, com cada jogador controlando um conjunto de peças.

RF-2: O sistema deve permitir que os jogadores movam suas peças de acordo com as regras do jogo Nine Men's Morris.

RF-3: O sistema deve destacar as peças que podem ser movidas a partir da posição atual do jogador.

RF-4: O sistema deve exibir um tabuleiro de Nine Men's Morris com 24 posições e 9 peças para cada jogador.

RF-5: O sistema deve notificar o jogador quando um movimento inválido for feito, com uma mensagem explicativa.

RF-6: O sistema deve notificar o jogador quando ele tentar mover uma peça que não é sua.

Requisitos não funcionais:
RNF-1: A interface gráfica deve ser intuitiva e fácil de usar, sem necessidade de treinamento prévio.

RNF-2: O sistema deve ter uma resposta de no máximo 1 segundo entre o clique do jogador e a atualização do tabuleiro.

RNF-3: O jogo deve ser totalmente jogável em diversas sessões, sem a necessidade de fechar e abrir de novo o jogo depois de uma partida.

RNF-4: O sistema deve funcionar corretamente em uma tela com resolução mínima de 1024x768 pixels.