Pm - Projeto Final

Artur Vaz - Davi Beltrao - Giovanni Tagliaferri

2017/1

1 Introdução

Ao longo do semestre fomos familiarizados com boas praticas de programação orientada a objetos, técnicas de modelagem de software(UML), padrões de projeto e assuntos como interface gráfica, tratamento de exceções e teste de software. Por fim, foi feito um trabalho de implementação do jogo Xadrez afim de colocar em pratica tudo que foi visto e desenvolver a pratica do trabalho em grupo.

2 Xadrez

Xadrez é um esporte/jogo competitivo de 2 pessoas em que o objetivo final é capturar o Rei do oponente. A partida é disputada em um tabuleiro 8x8 de casas claras e escuras, cada jogador controlar 16 peças com características diferentes.

T	C	B	Q	K	B	C	T
P	P	P	P	P	P	P	P
P	P	P	P	P	P	P	P
T	C	B	Q	K	B	C	T

- P = Peão: Movimenta uma casa para frente(duas na primeira vez) e captura pela diagonal.
- T = Torre: Move e captura horizontalmente e verticalmente sem limites.
- B = Bispo: Move e captura pelas diagonais sem limites.
- C = Cavalo: Move e captura em L.
- Q = Rainha: Agrega as regras da Torre e do Bispo.
- K = Rei: Assim como a rainha porem move apenas uma casa.

3 Implementação

Como ponto de partida foi utilizado um padrão de arquitetura de software chamado *Model-View-Controller* que separa a representação da informação da interação.

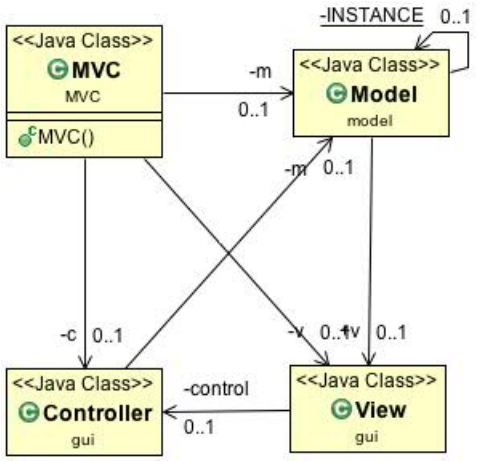

Figure 1: Diagrama de classes do MVC

3.1 Model

A função do *Model* é implementar a lógica e as estruturas de dados necessárias para o projeto. No contexto da nossa aplicação a estrutura principal é o tabuleiro, que é representado como uma matriz de posições e foi implementado como um **Singleton**, devido a necessidade de apenas uma instancia. Além das estruturas o *Model* possui uma referencia ao *View* já que é responsável por mandar as atualizações do estado para a Interface. O *Model* em nosso contexto tem a responsabilidade de verificar se a jogada é valida(foi usado a ideia de *States* para controlar a validade da jogada), verificar se a partida terminou, fazer a conexão do modo Multiplayer(seção 5) e sincronizar o estado entre os conectados, decidir o melhor movimento no modo versus AI(seção 4).

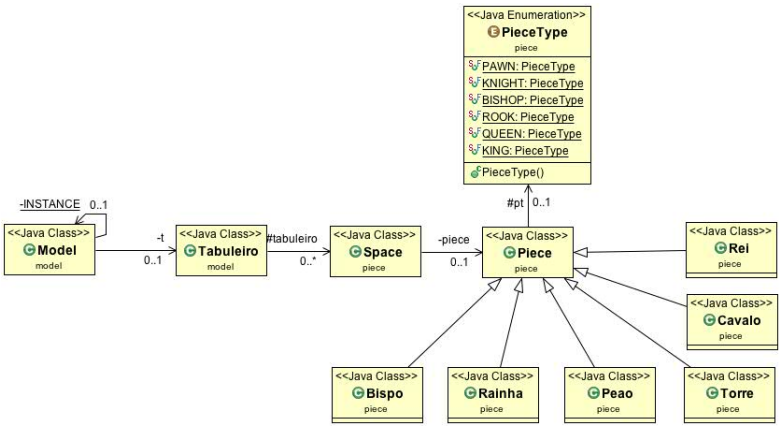

Figure 2: Diagrama de classes do model

3.1.1 Piece

A classe **Piece** implementa uma peça genérica, existente em um ponto bidimensional. As coordenadas são inteiros, cada peça possui informações sobre seu time, o tipo de peça (Enumeration com constantes indicando os tipos de peça de xadrez) e uma lista de pontos que são suas jogadas possíveis. A extensão desta classe é feita para determinar o tipo especifico de uma peça e sua movimentação específica.

3.1.2 Tabuleiro

O **Tabuleiro** é responsável por manter o estado do jogo, sua principal estrutura de dados é uma matriz 8x8 de **Space**, uma classe que contém uma única peça. Esta classe possui diversos métodos para organizar, computar jogadas e construção do estado inicial.
Possui também as classes dos jogadores em sua composição, podendo ser de dois tipos, o jogador humano ou o jogador de inteligência artificial, discutido na próxima sessão.

3.1.3 AI

A inteligência artificial(AI) foi desenvolvida com o objetivo de se poder jogar o xadrez com somente um jogador. Para isso, foi utilizado um algoritmo que utilizou de base o algoritmo MiniMax. O algoritmo MiniMax, resumidamente, funciona da seguinte forma: cria-se uma matriz em que todas as escolhas possíveis são armazenadas, e verificando essa matriz, retorna a melhor escolha possível. Dessa forma, o algoritmo criado para a inteligência artificial deste trabalho verifica para cada posição atual do tabuleiro e para as posições que cada peça pode mover qual o melhor movimento possível de se fazer para a dada rodada. Esse melhor movimento é determinado por algumas variáveis. Uma delas foi

o mapeamento do tabuleiro em que cada posição tem um determinado peso, e as posições centrais possuem peso maior do que as laterais pelo fato de que as posições centrais possuem maior vantagem no jogo. Outra variável foi o peso que cada peça possui. Isso se deve ao fato de que peças como o peão são menos valiosas do que peças como a rainha por possuírem maiores limites de movimentação. Assim sendo, a peça Rei possui maior score possível(infinito), pois se essa peça for comida o jogo termina. Esse algoritmo, então, deve testar todas as posições possíveis de cada peça ser movida para escolher qual o melhor movimento. Para isso foi utilizado o padrão de projeto Memento que é o responsável por salvar o estado da peça enquanto ela executa um possível movimento e depois retorna para a posição em que ela se encontrava. Além disso foi criada a classe AI-Player que é responsável por mapear as posições do tabuleiro e definir o peso das peças.

3.1.4 Socket

O Multiplayer foi implementado utilizando sockets, pontos de conexão entre redes, para enviar e receber dados. Em cada instância do jogo rodando, há internamente dele duas variáveis, uma contendo um **ServerSocket** e outra contendo apenas o **Socket**. O primeiro serve para criar o ponto de comunicação na rede, sendo apenas utilizando quando o jogador opta por ser o host da partida. Já o Socket é utilizando me ambos, é implementado um canal de comunicação que envia inteiros em pares entre os jogadores.
O inteiro enviado valerá entre 0 e 7 para falar em qual posição do tabuleiro uma peça foi selecionada e para onde ela será enviada e no fim do turno de um jogador, é enviada uma mensagem com um inteiro maior que 8 para o oponente, esta mensagem é um aviso que o outro socket precisa incrementar o contador de round para que o próximo jogador consiga mover sua peça.
A validação do movimento ocorre localmente, e é enviada a mensagem para o outro socket no momento que o movimento é registrado no jogo local. Há uma thread em execução que recebe os inputs da rede, esta thread foi necessária para que haja atualização em tempo real do tabuleiro.

3.2 View

O View é a interface visual do nosso programa e foi implementada usando a combinação dos pacotes **Swing** e **AWT** nativas do Java. A biblioteca swing provem elementos de interface que independem do sistema operacional usado, o awt provem elementos de interface dependentes do sistema.
A classe **View** é composta de duas subclasses extendidas de dois elementos do **Swing**. A classe **MenuText** foi utilizada para fazer os textos do menu do jogo, ela estende a classe **JLabel** do **Swing**, contento informações como posição e métodos para fazer a marcação de qual texto esta selecionado em amarelo. Há também a classe **MenuPanel** que possui que estende um **JPanel**, classe também da biblioteca do **Swing**, esta classe possui informações sobre sua posição relativa, informação necessária pois utilizamos esta classe na composição

do tabuleiro.
Para manter as janelas organizadas, foram utilizadas do pacote **AWT** as diversas formas de layout para que os elementos de interface não ficassem bagunçados. O layout mais importante que compões esta interface é o **CardLayout** que é utilizado para fazer a transição de diversas janelas presentes no jogo, como as telas de menu ou a tela do jogo.
As requisições para repintar são enviadas do model, geralmente é atualizado o tabuleiro logo após uma jogada.

Controller: Como o jogo funciona exclusivamente com inputs do mouse, o controller é uma implementação da classe **MouseListener** que pertence ao pacote **awt.event** que contem métodos para capturar a posição do click na interface. A partir da posição do click é enviado um sinal para o *Model* e com a sequencia de clicks é possível verificar a validade da intenção do usuário e assim executar a jogada.

4 Conclusão

Esse trabalho foi de extrema importância para o grupo aprender a utilizar as ferramentas gráficas swing e awt disponíveis no java, além delas foi utilizado o MVC, conceito aprendido em sala de aula, que ajudou a dividir o trabalho em três partes que foram divididas entre os integrantes para serem desenvolvidas em paralelo(de certa forma). Os problemas maiores foram em implementar o xadrez sem perder os conceitos de OO, que é importante para garantir uma boa qualidade de código apesar da maior dificuldade, por outro lado a implementação da parte gráfica e do controlador não foi problemático mesmo sendo a primeira vez. Nessa trajetória de implementação foi de grande ajuda os padrões de projeto para alguns pontos, seja no Singleton para o tabuleiro, State para fazer o Menu e Memento para recuperar a rodada anterior. De extra foi implementado dois modos, o Multiplayer e o Jogador contra AI que foi necessário aprender a parte de Sockets do java para fazer a comunicação de dois jogadores via IP e de desenvolver um sistema simples de melhor jogada para a AI, tudo isso sem nenhum interferir no outro.