Relatório do Projeto - SCC 0504

Integrantes do Grupo:
Andrey Lucas Esteves Garcia **NroUsp:** 10734290
Gabriel Muniz Morão **NroUsp:** 7236785

O que foi feito:

Foi utilizado como base para o projeto o código disponível na página da disciplina para a implementação na linguagem JAVA de um jogo de Xadrez. O projeto inicial continha a implementação através do método MVC (Model View Controller) do tabuleiro e suas classes necessárias para ele ser exibido na tela. Além disso o projeto continha implementações da classe controladora do tabuleiro e de uma classe abstrata do tipo peça. O projeto foi desenvolvido em ide NetBeans 10.0 e NetBeans 8.2.

A princípio fizemos apenas alterações em 5 das classes já pré implementadas, sendo essas alterações:

- **Na classe peça:** foi adicionado o método abstrato move_to(int x, int y); esse método é sobrescrito nas classes herdeiras de peça e serve para checar se o movimento executado é válido, e caso seja a peça mudará suas coordenadas para a nova casa onde será colocada.
- **Na classe Main:** apenas instanciamos uma variável do tipo model e chamamos o metodo addModel da classe TabuleiroController.
- **Na classe ModelTabuleiro:** foi modificada a função init(); onde foram adicionadas no vetor controlador de peças todas os outros tipos de peças, criadas a partir de outras classes que foram criadas para cada tipo de peça, como será descrito em outra seção deste relatório.
- **Na classe TabuleiroController:** foram criadas as variáveis “private Peca p”, “private Peca peca_clicada”, “private boolean flag”, “private int jogador” e “private boolean mat[][]”. Estas variáveis servem para, respectivamente, receber a peça (ou vazio) que se encontra no quadrante selecionado, salvar as informações da peça caso ela seja encontrada, salvar a informação de quem é a vez, informar qual jogador tem a vez e reproduzir o jogo numa matriz de booleanos. No método public void addModel(ModelTabuleiro model) foram instanciadas tais variáveis, com p e peca_clicada = null além de jogador = 1. Já na classe public void mousePressed(MouseEvent e), fazemos a verificação de que se há ou não peça no quadrante escolhido, utilizando o método findPeca(int x ,int y) de model. São verificados então o conteúdo das variáveis, como peças nulas e flag true, para verificar a vez de cada jogador. Se peca_clicada possuir um conteúdo não null, entramos na verificação de movimento e até mesmo verificações de “comer peça” e fim de jogo. Foram implementados nessa classe os métodos addmat() e printmat(), que inicializa a matriz para o jogo e a printa no console.
- **Na classe View:** foram adicionados 2 labels que encontram-se no canto inferior direito da janela para controle da vez do jogador e para indicar se algum jogador ganhou o jogo.

Além dessas alterações nas classes já existentes criamos novas classes **herdeiras** da classe abstrata peça sendo elas: Cavalo.java, Bispo.Java, Rei.Java, Rainha.Java e Torre.Java.

Todas essas classes sobrescrevem o método Draw, de acordo com a coordenada na imagem pecas.png que a peça em si representa, o método ToString(da classe Object), simplesmente para realizar prints no console indicando qual é essa peça e sua respectiva cor, e por fim todas as peças sobrescrevem o método boolean move_to(int x, int y, boolean mat[][]); que serve para realizar o movimento da peça, ele retorna true caso o movimento realizado seja válido, levando em consideração o tipo de movimento da peça e se não há nenhum obstáculo entre a peça e o destino, para isso a matriz de booleanos, e false se o movimento realizado for inválido, alterando as coordenadas da peça, para as novas, apenas no caso true. A verificação dos movimentos é baseada nos movimentos específicos de cada tipo de peça do jogo de xadrez.

Dessa forma nosso programa usa o conceito de **herança** específico de linguagens orientadas à objeto, que no caso da linguagem JAVA, onde as peças específicas recebem da classe mãe (peça) suas características de cor e coordenadas.

Já o conceito de **Polimorfismo** pode ser observado na sobrescrita dos métodos das classes filhas perante a classe mãe. Ao verificar se há uma peça no quadrante, caso uma peça seja encontrada, o programa verifica qual o tipo (peão, rei, etc) e chama o método draw (sobrescrito) daquela classe filha. De forma análoga, são chamados os métodos toString e move_to deste tipo de peça encontrado. Não há assim uma verificação do tipo codificada durante a execução, isso se dá através do Polimorfismo e os métodos sobrescritos das classes filhas perante a classe mãe (abstrata) .

Tal conceito é visto explicitamente na classe TabuleiroController, especificamente no método subscrito public void mousePressed(MouseEvent e); onde a função do objeto Model retorna um tipo abstrato de peça, que é guardado na variável P, através dessa variável, evocamos o método move_to(int x, int y, boolean mat[][]), passando as coordenadas para onde queremos que o movimento se realize e a matriz de controle do jogo, nesse momento o **Polimorfismo** entra em ação, chamando o método específico da classe que p recebeu, neste método o movimento é checado e se caso válido ele é refletido visualmente no tabuleiro. Um exemplo é, se eu clico em um cavalo e clico logo no quadrado à sua frente, o movimento não ocorre e um aviso de “Movimento Inválido” é colocado no console, visto que o cavalo só pode se mover em “L”. Caso a peça clicada fosse um peão e não um cavalo, o método chamado seria o move_to da classe peão e o movimento seria validado, visto que o peão pode andar para frente.

Além desse caso o **Polimorfismo** é usado também no método toString, que é usado para exibir no console o que está acontecendo no tabuleiro, no caso em específico de o rei ser comido por outra peça, caso isso ocorra, uma verificação é feita a partir do retorno desse método e os labels da tela de jogo mudam, indicando que uma das equipes saiu vencedora da partida.

Considerações finais:

Futuramente, essa mudança de labels pode ser modificada para uma mudança de telas indicando qual equipe saiu vencedora e perguntando para o usuário se ele quer jogar de novo ou sair do jogo, porém como esse não foi o foco dessa parte do projeto prático,

essa feature não está implementada, é apenas uma ideia. De mesma forma, exceções podem ser tratadas, assim como utilização de collections, threads e até mesmo pensar em padrões de modelagem e projetos. Assim como dito anteriormente, são apenas ideias que podem vir a ser implementadas em um futuro.