Dou Shou Qi - Padrões de Projeto

Ivens Diego Müller[1]

[1]CEAVI - Universidade do Estado de Santa Catarina(UDESC) Ibirama - SC - Brasil

ivens.muller@edu.udesc.br

1. Informações Gerais

Jogo Dou Shou Qi implementado utilizando a linguagem de programação Java para a disciplina de Padrões de Projeto para o curso de Bacharelado em Engenharia de Software no Centro de Educação Superior do Alto Vale do Itajaí na Universidade do Estado de Santa Catarina(UDESC). Foi utilizado o padrão de arquitetura MVC e os padrões de projeto Abstract Factory, Builder, Command, Observer e Singleton.

2. Model-View-Controller(MVC)

O padrão de arquitetura MVC foi implementado no jogo. Implementou-se 3 pacotes.

2.1. Model

Contém todas as classes de modelo do sistema, como peças, animais, fábricas abstratas(Abstract Factory) e as classes do Builder.

2.2. View

Contemos apenas 3 classes, uma delas é a Tabuleiro, que herda de um JFrame e é a view onde contém a JTable que é utilizada para implementar o tabuleiro.

A outra, chamasse TabuleiroRenderer, que é a classe responsável renderização do JTable, desenhando as imagens nas suas posições.

A terceira, chamada de TabuleiroTableModel, é a classe responsável por buscar os dados de cada posição do controller e implementa o método do Click no mouse para realizar a movimentação das peças (chamando o controller para validade).

2.3. Controller

No controller, foi implementado a classe TabuleiroController, que tem todas as regras de criação dos objetos, movimentação das peças e implementação do tabuleiro em si. Também, há a interface do Command e todas as classes do padrão command.

3. Observer

O padrão observer foi implementado da seguinte maneira: Criado a interace ObservadorTabuleiro no pacote Model, onde o TabuleiroTableModel implementa essa interface. O controller tem uma lista de observadores do tabuleiro que, quando o TabuleiroTableModel cria um objeto do controller, ele se cadastra como observador. A criação do controller e o cadastramento como observador acontecem no construtor do TabuleiroTableModel.

Essa interface tem 5 métodos. Que são chamados pelo controller. Um é chamado para notificar o carregamento do tabuleiro, outro para enviar mensagens, notificar alteração no tabuleiro, notificar a troca da imagem de uma peça e notificar o fim do jogo.

4. Abstract Factory

No pacote Model, foram criados duas Fábricas Abstratas, uma que é chamada de FabricaDePeca, essa fábrica possui dois métodos abstratos, um utilizado para cria uma peça para o jogador 1 e a outra para o jogador 2. Ambos os métodos recebem como parâmetro uma linha e uma coluna e retornar um objeto do model Peca. As fábricas concretas são: FabricaDeArmadilha, FabricaDeCachorro, FabricaDeElefante, FabricaDeGato, FabricaDeLeao, FabricaDeLeopardo, FabricaDeLobo, FabricaDeRato, FabricaDeTigre e FabricaDeToca.

A outra chamasse FabricaDeObjetoTabuleiro, que tem um método abstrato chamado criaObjetoTabuleiro que retorna um ObjetoTabuleiro da classe Lago. A fábrica concreta é: FabricaDeLago.

5. Builder

Há 2 Builder's implementados. Um para criar as pecas de cada jogador, e outro para criar os objetos do tabuleiro. O Builder das peças, tem a classe abstrata BuilderJogador, que tem os métodos para criar cada tipo de peça, Tem duas classes concretas chamadas ConcretBuilderJogador1 e ConcretBuilderJogador2 que implementam os métodos abstratos. A classe DiretorJogador, recebe um Builder por parâmetro e chama o método de criação das peças. Esses métodos implementados de criação das peças, chamam o abstract factory para criar a peça desejada. O Builder dos objetos do tabuleiro tem uma classe abstrata BuilderObjetoTabuleiro que tem um método abstrato para criar Lagos, e tem uma classe chamada ConcretBuilderTabuleiro que implementa esse método, chamando a fábrica de lagos passada por parâmetro para criar um novo lago. A implementação do builder pode ser vista no método adicionaPecasNoTabuleiro dentro da classe TabuleiroController no pacote Controller na linha 83.

6. Command

O padrão command, possui uma interface chamada Command que fica no pacote do Controller, essa interface tem um método chamado execute que recebe uma matriz de objetos do tabuleiro, um animal e um objeto padrão. Os comandos criados as classes AndarUmaCasaParaCima, AndarUmaCasaParaBaixo, AndarUmaCasaParaEsquerda, AndarUmaCasaParaDireita, PularLagoParaCima, PularLagoParaBaixo, PularLagoParaEsquerda e PularLagoParaDireita. Todas essas classes implementam a interface Command. Dentro do método execute, na posição atual do animal vindo por parâmetro é setado o objeto padrão, na posição definida pelo tipo do command é setado o animal e é alterado a posição atual da classe do Animal. Isso pode ser visto no método movimentaPecaParaPosicao na classe TabuleiroController, iniciando na linha 247.

7. Singleton

O padrão singleton foi implementado na View Tabuleiro, onde a classe Tabuleiro tem um objeto de si mesma chamado instance, um construtor privado que faz a criação do Frame, e um método síncrono chamado getInstance() que retorna um Tabuleiro, que verifica quando o objeto instance está nulo, ele chama o construtor privado e sempre no final retorna o instance. Isso pode ser visto na classe Tabuleiro, no pacote View, onde o método getInstance() começa na linha 29.