Apresentação dos Padrões de Projeto Aplicados no Desenvolvimento do Jogo de Tabuleiro: Haru Ichiban

Denilson Laucsen da Rosa[1], José Carlos Bernardes Brümmer[2]

[1]Departamento de Engenharia de Software – Centro de Educação Superior do Alto Vale do Itajaí (CEAVI) – Universidade do Estado de Santa Catarina (UDESC)
89.140-000 – Ibirama – SC – Brasil

denilsonlaucsen@gmail.com, jcbebr@gmail.com

Abstract. *This paper presents the development of an academic work in Design Pattern's area. In that, a Japanese board game called Haru Ichiban was implemented. There are among the constraints of development: Language (Java), and patters (MVC, Observer, Singleton, Abstract Factory, Builder, Command).*

Resumo. *Esse artigo apresenta o modo de desenvolvimento de um trabalho acadêmico na área de Padrões de Projeto. No qual, foi implementado um jogo de tabuleiro japonês chamado Haru Ichiban. Estão entre as restrições do projeto: A linguagem de programação (Java), e os padrões (MVC, Observer, Singleton, Abstract Factory, Builder, Command).*

1. Introdução

A elaboração do projeto em questão se baseia na implementação do Haru Ichiban. O qual, é um jogo de tabuleiro japonês, traduzido para "O primeiro vento da primavera". Neste jogo, dois jardineiros disputam através de diferentes estratégias, pela maior pontuação. O vencedor é denominado de Jardineiro Imperial.

As regras do Haru Ichiban não serão discutidas, afinal o foco deste trabalho foi identificar possíveis usos para os Padrões de Projetos, suas respectivas justificativas, e posteriores aplicações. Portanto, abaixo seguem as explicações idealizadas e as referências em código para as mesmas.

2. Padrões de Projetos:

2.1. Padrão MVC

O projeto foi desenvolvido estruturalmente de acordo com o este padrão. De modo em que as classes estão todas divididas nos pacotes model, view e control, seguindo as determinações do mesmo.

Ele foi planejado no código com a finalidade de aprimorar sua manutenção e escalabilidade. Afinal, por se tratar de um projeto amplo, e que será divido por diferentes entregas, é esperado um crescimento considerável de classes, então é preciso haver esta organização padronizada.

2.2. Padrão Observer

Encontrado em todas as classes e interfaces do pacote br.udesc.ppr.haruichiban. control.observer. Foi utilizado diretamente em conjunto com o padrão MVC, com a finalidade de notificar a camada de visualização as mudanças ocorridas nos observados.

2.3. Padrão Singleton

Este padrão foi aplicado na classe GameController, a qual possui o método público e estático getInstance(), que retorna uma instância da própria classe, assim como o padrão em questão determina.

Esta aplicação foi pensada para facilitar o controle de fluxo do jogo, visto que a partir de diversos pontos do código precisa-se realizar a modificação das rodadas, assim como a obtenção de determinados controladores de outros objetos do pacote de visualização, são eles: RedHandController, YellowHandController e BoardController.

Outra funcionalidade desta classe, também referente ao controle de jogo, é a obtenção do jardineiro sênior. Uma vez esta configuração é mantida através do andamento das rodadas.

Por fim, e por ser a classe de controle de fluxo, há a funcionalidade aplicada a um novo jogo, em que ocorre a anulação da instância estática do padrão, fazendo com que na próxima vez que o método getInstance() for chamado será criado uma nova instancia da classe, determinando assim um novo jogo.

2.4. Padrão Abstract Factory

Este padrão foi aplicado nas instâncias de flores do jogo. Esta definição foi pensada pelo fato de existir, para cada flor de zero a oito, uma imagem diferente. Sendo que, a flor zero é a flor sem numeração. Portanto, foi criada a classe abstrata FlowerFactory contendo os métodos: buildFlower00(), buildFlower01(), buildFlower02(), buildFlower03(), buildFlower04(), buildFlower05(), buildFlower06(), buildFlower07() e buildFlower08(). Todos eles possuem como retorno um objeto da classe Flower.

A classe FlowerFactory foi estendida pelas classes RedFlowerFactory e YellowFlowerFactory. Portanto, para cada método da classe RedFlowerFactory é retornado uma instancia de uma flor vermelha correspondente ao número do seu método. Assim também ocorre na classe YellowFlowerFactory, porém com as flores amarelas.

Finalmente, é utilizado no baralho de cada jogador, ou seja, como um atributo na classe abstrata Deck. E posteriormente nas classes RedDeck e YellowDeck em seus construtores. Construindo o baralho referente a cada jogador pela cor determinada.

2.5. Padrão Builder

Projetado com a finalidade de montar o tabuleiro do jogo com diferentes dificuldades, ou seja, diferentes organizações de tabuleiros. Ele é encontrado no pacote br.udesc.ppr.haruichiban.control.builder, sendo que foi implementado o concrete builder dado pela classe EasyBuilder. Esta, por sua vez estende da classe abstrata Builder, a qual possui uma instancia de um tabuleiro Board utilizada posteriormente na classe BoardController.

O processo de criação do tabuleiro é efetivado pelo Director, chamado no construtor da classe BoardController.

2.6. Padrão Command

Seu uso foi realizado no tabuleiro, mais especificamente durante a movimentação realizada pelo jardineiro júnior (Haru Ichiban).

A classe abstrata de comandos é a BoardCommand, encontrada no pacote br.udesc.ppr.haruichiban.control.command. Bem como a classe BoardCommandInvoker, responsável pela execução dos comandos.

Os comandos existentes são divididos em dois grupos de quatro classes (comandos) cada, o primeiro grupo realiza uma busca em recursividade de determinada peça no tabuleiro, sendo um comando para cada direção, e o segundo grupo realiza a movimentação de peças no mesmo, igualmente com um comando para cada direção.

Seu uso efetivo é encontrado na classe BoardController, mais especificamente nos métodos STAGE06() e STAGE07(). Sendo que o primeiro realiza a busca citada anteriormente, e o segundo a movimentação.

Apesar de não haver a implementação dos métodos undo() e redo() o padrão foi útil no projeto para simplificação dos algoritmos implementados e execução recursiva dos mesmos. Além da organização de código e possível implementação futura de um log de execução de comandos, afinal futuramente haverão jogadores em diferentes máquinas. Portanto necessita-se um controle de movimentos do tabuleiro.

3. Material utilizado

Os materiais utilizados durante o desenvolvimento foram baseados nos conteúdos vistos em sala de aula. Bem como a bibliografia repassada no plano de ensino da disciplina.

4. Referências

Gamma, E. (2000). “Padrões de Projeto: soluções reutilizáveis de software orientado a objetos”. Bookman editora.