Descrição dos Padrões de Projeto implementados no projeto DouShouQi

Kevin Kons[1]

[1]Departamento de Engenharia de Software
Universidade Estadual de Santa Catarina (UDESC) – Ibirama, SC – Brazil

kevin.kons@hotmail.com

Abstract. *This meta-paper describes the utilization of the Design Patterns implemented in the second delivery of the game DouShouQi implemented to the Design Patterns subject. The Design Patterns applied were: Visitor, Strategy e State.*

Resumo. *Este meta-artigo descreve a utilização dos Padrões de Projeto implementados na segunda entrega do jogo DouShouQi implementado para a disciplina de Padrões de Projeto. Os padrões aplicados foram: Visitor, Strategy e State.*

1. Visitor

Neste trabalho o Visitor possui o objetivo de retornar as peças mortas por um time. A interface Visitor está localizada no pacote “model.time.visitor” e possui o método “visit” que recebe uma peça a ser visitada e um método “getValue” que retorna o resultado das visitas em um time. No mesmo pacote está a classe “BuscaPecasAtacadasVisitor” que adiciona as peças adversárias mortas pela peça visitada em uma lista que será retornada no método “getValue”. Na linha 73 da classe “Time” do pacote “model.time” está o método “accept” que recebe um visitor como parâmetro que é utilizado para visitar cada uma das peças de um time.

2. Strategy

Este padrão foi implementado com o intuito de definir uma família de algoritmos utilizados pela classe “Time”. A interface “StrategyTime” localizada no pacote “model.time.strategy” defini a interface a ser implementada pelas classes concretas de estratégia. Na interface é declarado o método “calcular” que retorna um objeto diferente dependendo da classe que implementa essa interface, como parâmetro é recebido um objeto time. Existem duas classes concretas de estratégia de time, ambas localizadas no pacote “model.time.strategy” a primeira é “CalcPontuacaoStrategy”, essa classe defini um algoritmo que retorna a pontuação de cada time sendo essa calculada com base na soma das forças das peças adversárias que foram mortas pelo time que possui a estrátegia. A instanciação dessa classe ocorre no método “atualizaPontuacao” da classe “TabuleiroController” localizada no pacote “controller”. A segunda classe de estratégia é “IdentificaPecasAtacadas” que tem como objetivo retornar uma string com o nome de todas as peças adversárias mortas pelo time. A instanciação dessa classe está no método “jogoVencido” que é declarado na linha 173 da classe “TabuleiroController”. Na classe “Time” do pacote ”model.time”na linha 79 está o método “setStrategy” que recebe como parâmetro uma estrátegia que será utilizada pelo time, já na linha 83 é declarado o método “calcularStrategy” que retorna o resultado da execução de um algorítmo de estratégia.

3. State

O padrão State defini o estado de um controller de tabuleiro, o controller possui dois estados: Não Seleção que ocorre quando não há nenhuma peça selecionada e Seleção que ocorre quando uma peça é selecionada. No pacote “controller.state” possuímos a classe abstrata que defini o State, ela possui uma referência da classe que possui o estado, um CommandInvoker e um método utilizado para definir um novo estado. O estado de Não seleção está implementado na classe “NaoSelecao” do mesmo pacote de sua classe pai, do estado de não seleção é possível apenas selecionar uma peça, levanto o controller à um estado de seleção. Sendo este definido na classe “Selecao”, com uma peça selecionada é possível selecionar uma peça diferente mantendo o controller em estado de seleção, movimentar a peça selecionada ou atacar uma peça adversária, o que leva o controller à um estado de não seleção. A chamada do método “proxEstado” que define o novo estado está ocorre na classe “TabuleiroController” no método “tabuleiroClicado” localizado na linha 48.