UFMG

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UNIVERSIDADE FEDERAL DE MINAS GERAIS, BRASIL

(UFMG)

Blizzard Heroes

Uma versão de Super Trunfo baseada no mundo Blizzard

Belo Horizonte, 15 de Maio de 2017

Autores:

Daniel Vieira da Silva Cruz & Eugênio Pacceli Reis da Fonseca

1 - Introdução

No segundo trabalho prático de Programação Modular, matéria lecionada por Douglas G. Macharet pelo Departamento de Ciência da Computação, da Universidade Federal de Minas Gerais, foi proposta a implementação de uma versão customizada do jogo de cartas “Super Trunfo”.

O Super Trunfo é um jogo de cartas que consiste em obter todas as cartas em jogo dos adversários. Isso é feito através de rodadas, onde um jogador pivô seleciona uma característica de carta como critério de comparação, a ser usado para definir a vencedora da rodada (cada jogador saca uma carta para a jogada). O melhor valor ganha e o dono da carta recebe todas as cartas da rodada, além de ser o jogador pivô da próxima.

Existe uma carta especial, o Super Trunfo, que automaticamente ganha a rodada quando sacado, exceto se outro jogador tiver sacado uma carta de categoria ‘A’, pois essa categoria trunfa o Super Trunfo (mas são cartas comuns para toda jogada que não envolve o Super Trunfo).

2 - Apresentação da implementação

O programa foi escrito para Java 8, testado na JDK 8u131, da Oracle.

Pacote edu.blizzardheroes.app

Esse pacote contém a classe Main, responsável por inicializar o ambiente do JavaFX - implementando a interface Application do pacote JavaFx - e carregar a View principal do jogo através do padrão *Dependency Injection* - passando o endereço do FXML a ser carregado para uma instância de FXMLLoader -. A classe Main é o ponto de entrada do programa.

Pacote edu.blizzardheroes.assets

Este pacote possui apenas recursos de imagem para construção da tela principal do jogo. Dentro dele, existem subpacotes para cada categoria de carta, facilitando o load no momento construção das imagens das cartas:

edu.blizzardheroes.assets.cards.assassins

Possui as cartas da categoria assassino (A).

edu.blizzardheroes.assets.cards.warriors

Possui as cartas da categoria guerreiro(B).

edu.blizzardheroes.assets.cards.supports

Possui as cartas da categoria suporte(C).

edu.blizzardheroes.assets.cards.specialists

Possui as cartas da categoria especialista(D).

Pacote edu.blizzardheroes.gameplay

Este pacote possui uma única classe *GameTable* que corresponde à estrutura em que o jogo de fato ocorrerá. Esta classe é responsável pela execução dos turnos, onde cada jogador joga uma carta e o pivô seleciona o atributo de comparação. Ao final do turno uma carta é vencedora.

Uma observação aqui é importante, pois encontramos variações para o empate/desempate de cartas. Chegamos a um consenso de simplificar o jogo e decidimos nestes casos, dar a vitória à primeira carta dentre as que estiverem empatando. Devido ao balanceamento feito nas cartas, isso não afeta a dinâmica nem a dificuldade do jogo.

Pacote edu.blizzardheroes.gui - Extra

O pacote GUI contém o arquivo FXML que descreve a interface tela-cheia que aparece para o usuário interagir e a classe controladora do mesmo, pois a plataforma JavaFx faz uso do *padrão de arquitetura de software* chamado MVC (*Model-View-Controller*), que prevê a separação entre interface, controlador e modelo, com interações bem definidas entre eles - o controlador processa as entradas do usuário e do ambiente, atualiza o modelo, que é mostrado na interface -.

O JavaFx provê o pacote de classes dos objetos que são desenhados na tela, pacotes de desenho e manipulação de imagens, o rendering do FXML na tela e a ligação entre ele e a lógica do programa através do controller definido pelo usuário e instanciado e configurado por um *FXMLLoader*, do JavaFx.

Há também, no pacote, um enum de estados que representa os estados possíveis do jogo, manipulados no controlador através da interação do usuário com a interface.

A interface possui estados diferentes, cada um representando um estado do jogo e da tel. Os estados são controlados no método *sync*, que é acionado toda vez que o usuário aperta o botão de interação. O *sync* executa uma ação dependendo do estado atualizando o jogo, a interface e passando para próximo estado.

A interface é composta por uma legenda, para descrever ao usuário o que está acontecendo, do espaço de cada jogador e local para eles mostrarem as suas cartas (um painel com título e dois *Label*, com um *ImageView* do lado, que é onde a carta aparece, ambos dentro de um painel horizontal transparente, *HBox*), de um botão de interação com o jogador (*Button*) no espaço dele e de um diálogo perguntando por entradas do jogador. Isso tudo dentro de uma grade (*GridPane*), exceto o diálogo, que aparece em janela separada.

O *MainWindowController* interage com o usuário através de um botão, onde ele pode manifestar sua intenção, além de um *Dialog*, que aparece quando é a vez do jogador de escolher a carta pivô.

O *Dialog* é uma janela de entrada de dados que aparece na frente da interface, provida pelo JavaFx, implementa um *Decorator* para que possa ser configurada (texto, imagem, escolhas, cabeçalho, título da janela).

As cartas e a interface foram feitas à mão, com assets do jogo *Heroes Of The Storm*, da *Blizzard*.

Pacote edu.blizzardheroes.model.actors

Neste pacote constam os atores do jogo. Foi implementada uma classe abstrata *Player*, onde garantimos que todo jogador tenha cartas e um nome para ser identificado na interface gráfica.

Extendendo a classe Player temos a especialização em *HumanPlayer* e *ComputerPlayer* que diferem o jogador humano (Jogo Single Player) dos jogadores artificiais.

Pacote edu.blizzardheroes.model.cards

Neste último pacote, consta a unidade atômica de carta *Card* que possui os atributos relevantes para o jogo.

Além desta classe, criamos dois Enums fundamentais: o *CardCategory* e o *CardAttribute*. O primeiro serve para definirmos as categorias das classes, que é importante no jogo pois apenas categorias 'A' podem ganhar do super trunfo. A segunda serve para definir no turno, qual atributo será utilizado para comparação.

E por fim, a classe *Deck* realiza as operações de distribuição das cartas. Ela é responsável por inicializar as 32 cartas e distribuí-las de maneira aleatória entre os jogadores da partida.

4 – Conclusão

Este trabalho foi ainda melhor que o primeiro, pois além de melhorarmos nossa capacidade de modelagem e *design* de software, fizemos nosso melhor para

implementar um extra que em nossa perspectiva se aplica bem a um jogo, que é a interface gráfica.

Mantivemos a utilização dos conceitos do paradigma de orientação a objetos, aprendidos em sala de aula e nos estudos, buscando seguir metodologias como os princípios SOLID.

5 – Bibliografia

- Documentação e especificação da API do Java 8, Oracle < https://docs.oracle.com/javase/8/docs/api/ >
- Java - Como Programar - 10ª Ed. 2016, Prentice Hall, Deitel
- Trabalho Prático 2 – Cartas na mesa , Prof. Douglas G. Macharet, disponibilizado no moodle da disciplina e no diretório raiz desse trabalho.
- http://us.battle.net/ - Site da Blizzard onde obtemos recursos de imagem.