ISEL / ADEETC

Licenciatura em Engenharia Informática e Multimédia

Produção de Conteúdos Multimédia

Aula de Laboratório de Produção de Conteúdos Multimédia

Jogo de Cartas - Blackjack

Rui Jesus e João Beleza

Introdução

Este trabalho visa a introdução e a familiarização com a linguagem de programação JavaScript. O objetivo é o desenvolvimento do **jogo de cartas Blackjack**. Cada aluno/grupo tem de implementar os métodos/funções que estão por fazer no código fornecido pelo docente. Os alunos têm liberdade para fazerem alterações desde que não se afastem muito da estrutura.

Em baixo é apresentado um *link* para o *site* do w3schools que deve ser consultado durante o desenvolvimento do jogo.

http://www.w3schools.com/

Objetivos

O projeto fornecido pelo docente 3 ficheiros: "blackjack_oop.html", "blackjack_manager.js" e "blackjack_object.js". O primeiro implementa a interface com o utilizador ("View"), o segundo faz a interface entre a interface utilizador e o objeto "blackjack" ("Controller") e o terceiro contém o código do objeto (Model). Esta implementação segue o padrão de design MVC (Model, View, Controller) que permite decompor a aplicação em várias componentes, mais adequado para a manutenção e reutilização do código. A figura 1 ilustra o padrão de design MVC.

No final da primeira aula o aluno terá de ter, pelo menos, o objeto "blackjack" implementado, isto é, o aluno deverá fazer o código dos métodos do objeto. No final da segunda aula o aluno deverá ter a aplicação implementada, isto é, as funções do ficheiro "blackjack_manager.js". O **código completo do jogo Blackjack** tem de ser entregue até ao dia **29 de Outubro de 2017** utilizando a plataforma Moodle.

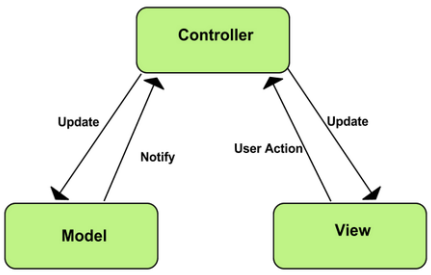

Figura 1. Componenentes do MVC.

Blackjack

1. **Objetivo**: somar 21 pontos com as cartas na mão.
2. **Elementos participantes**: o ***dealer*** que dá as cartas e o **jogador** que joga contra o ***dealer***.
3. **Pontuação das cartas**: as cartas do 2 ao 9 pontuam com o valor marcado na carta (2 a 9 pontos); o Rei, a Dama e o Valete valem 10 pontos e o Às pode ter o valor de 1 ou 11 (o jogador escolhe como lhe dá mais jeito).
4. **Condições iniciais**: são dadas duas cartas ao *dealer* e ao jogador. A 2ªcarta do *dealer* é voltada para baixo (não é visível qual é a carta).
5. **Dinâmica do Jogo**: após a 2ªcarta, o jogador pode passar o jogo ao *dealer* ou pedir cartas mais cartas. Se a pontuação das cartas do jogador exceder 21 pontos, o jogador perde imediatamente. Se a pontuação é 21, o jogador ganha imediatamente. Caso o jogador passe o jogo para o *dealer*, este joga até ter 21 pontos ou exceder a pontuação do jogador. Se o *dealer* exceder a pontuação do jogador, ganha, se rebentar, perde.
6. **Fim do jogo**: (1) jogador faz 21 pontos; (2) jogador rebenta; (3) *dealer* faz mais pontos do que o jogador; (4) *dealer* rebenta.

Trabalho Laboratorial

Objeto "blackjack"

1. Unzip o ficheiro "blackjack.zip" e crie um novo projeto no IntelliJ IDEA com os ficheiros fornecidos. Faça "NEW->Projeto from Existing Sources".

2. Implemente os seguintes métodos do objeto "blackjack":
   a. `new_deck()`

      Este método retorna um *array* com as 52 cartas representadas por números de 1 a 13 para cada naipe. Os números 11, 12 e 13 representam as figuras (Rei, Valete e Dama).

   b. `shuffle()`

      Este método baralha o *array* de cartas construído no método anterior e retorna um novo *array* baralhado.

Deve criar um *array* de índices de 1 a 52 (*for*). De seguida deve fazer um outro *for* que em cada ciclo (52) faça o sorteio (Math.random) de um índice. Este índice é usado para ir buscar uma carta ao baralho. Esta carta é inserida num fim de outro *array* com as cartas baralhadas. Não esquecer de remover o índice sorteado do *array* de índice.

c. `get_cards_value()`

Este método conta o valor de um *array* de cartas (*dealer_cards* ou *player_cards*) de acordo com as regras do blackjack. As cartas do 2 ao 9 pontuam com o valor marcado na carta (2 a 9 pontos). O Rei, a Dama e o Valete valem 10 pontos e o Às pode ter o valor de 1 ou 11 (o jogador escolhe como lhe dá mais jeito). Retorna os pontos resultantes.

d. `get_game_state()`

Este método verifica se a pontuação das cartas do *dealer* e do *player* permitem terminar o jogo (rebentar ou 21) e se alguém ganhou. Atualiza a variável "state" (membro da classe Blackjack) com o estado atual e retorna essa variável.

e. `player_move()`

Este método vai buscar a próxima carta ao baralho e coloca-a no *array* de cartas do *player*. Retorna a variável "state" atualizada executando o método `get_game_state()` que atualiza o estado do jogo após a jogada do *player*.

f. `dealer_move()`

Este método é igual ao anterior mas agora para o *dealer* (a nova carta deve ser colocada no *array* de cartas do *dealer*.

3. Implemente as funções do ficheiro "blackjack_manager.js"

a. `new_game()`

Esta função cria uma instância da classe "blackJack", executa duas jogadas do *dealer* e uma do *player*. Não se esqueça que a 2ª carta do *dealer* deve aparecer voltada para baixo. Por exemplo, deve substituir a apresentação da 2ªcarta do *dealer* pelo caracter "X". Finalmente, são inicializados os botões da interface utilizador.

b. `update_dealer()`

Esta função verifica se o estado do jogo é "gameEnded". Em caso afirmativo, pede à classe "blackJack" as cartas do *dealer*. Verifica se o *dealer* ganhou ou perdeu. E começa a construir uma *string* para mostrar as cartas acrescentando à *string* informação se o *dealer* ganhou ou perdeu. Finalmente atualiza a *string* no elemento HTML associado ao *dealer* e executa a função `finalize_buttons()`.

c. `update_player()`

Esta função pede à classe "blackJack" as cartas do *player* e começa a construir uma *string* para mostrar as cartas. A seguir, verifica se o estado do jogo é "gameEnded" e se o *player* ganhou de modo a acrescentar à *string* informação se o utilizador ganhou ou perdeu. Depois atualiza a *string* no elemento HTML associado ao *player* e executa a função `finalize_buttons()`.

d. `dealer_new_card()`

Esta função executa o método da classe "blackJack" que realiza a jogada do *dealer*, atualiza o *dealer* e retorna o estado do jogo.

e. `player_new_card()`

Esta função executa o método da classe "blackJack" que realiza a jogada do *player*, atualiza o *player* e retorna o estado do jogo.

f. `dealer_acaba()`

Esta função chama o método `get_game_state()` da classe "blackJack" e coloca a "**true**" a variável "DealerTurn da classe "blackJack". Depois é criado um ciclo até que o jogo termine (**state.gameEnded**). Nesse ciclo, é atualizado o *dealer*, realizada uma jogada do *dealer* e atualizado o estado do jogo em cada iteração.

4. Melhore a interface utilizador (opcional).
5. Acrescente as regras utilizadas em alguns casinos que não estão incluídas nesta versão simplificada (opcional)