Relatório do Problema 2: Jogo de Cartas BlackJack

Anésio Sousa dos Santos Neto

Departamento de Tecnologia
Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n, Novo Horizonte – 44036-900
Feira de Santana – BA – Brasil

anesios98@gmail.com

Abstract. *This report describes all the steps for the development of a card game called BlackJack, via console (UI). Using the MVC architecture pattern and the design pattern Facade along with data structures to optimize access to the data. The entire project was developed by two people, using the git versioning tool.*

Resumo. *Este relatório descreve todos os passos para o desenvolvimento de um jogo de cartas chamado BlackJack, via console (UI). Utilizando o padrão de arquitetura MVC e o design pattern Facade junto à estruturas de dados para otimização do acesso aos mesmos. Todo o projeto foi desenvolvido em dupla, utilizando a ferramenta de versionamento git.*

1. Introdução

A cidade americana de Las Vegas do estado de Nevada, é considerada a cidade com maior taxa de crescimento do país. Uma boa parcela disso se dá ao fato dos seus luxuosos cassinos, que atraem pessoas dos mais diversos lugares de todo o mundo, que querem faturar uma grana extra. Mauricinho Rico foi uma dessas pessoas atraídas pela cidade. Após voltar de uma luxuosa viagem da cidade, ele decidiu que queria ganhar dinheiro do mesmo modo dos cassinos. Por isso, Mauricinho investiu todo o dinheiro que ganhou contratando a empresa Jr para criar um jogo de BlackJack personalizado. Nós que participamos da empresa, devemos criar o software nos baseando dos pedidos do cliente. O sistema deve ser feito em uma linguagem de programação atual, utilizando as melhores estratégias possíveis, para uma melhor performance do mesmo.

2. Fundamentação teórica

Para criar o sistema foi necessário reunir algumas informações vitais para o cumprimento do que foi proposto, escolhendo formas e ferramentas para um melhor desenvolvimento do produto. Ferramentas como: linguagem a ser utilizada, padrões de projeto a serem utilizados e algoritmos de ordenação de dados.

2.1. Linguagem de programação

É inegável que atualmente para fins de trabalho e estudos linguagens orientadas a objetos (POO), como Java e php tem sido comumente mais utilizadas do que linguagens procedurais como o C e FORTRAN. Uma grande parte disso se dá ao fato de que a

POO combina práticas comprovadas e testadas da forma mais eficiente possível. A relação com o entendimento do mundo real torna a lógica de programação mais simples, além de deixar códigos de forma que facilita a manutenção. Existem 4 pilares de linguagens orientadas a objetos: Abstração, Encapsulamento, Herança e Polimorfismo. Já que modelagem de objetos é dada partindo de conceitos reais, a abstração é gerada ao ser pensado o que dado objeto representará no sistema, ou seja, sua identidade, responsabilidades e ações. Quando ele não conseguir realizar suas responsabilidades por si próprio, é então necessária contribuição de outros objetos. O encapsulamento vem de suprimir as propriedades dos objetos de uma forma em que eles se transformem em uma espécie de caixa preta. Evitando assim acesso direto ao objeto, dando uma segurança a integridade dos dados. A herança vem do reuso de código. Um exemplo de fácil entendendimento é a questão da herança genética, onde filhos herdam características de seus pais e tem características adicionais próprias. E por fim, o polimorfismo. Ele anda lado a lado com a herança, ele consiste na alteração do funcionamento de um método (ação de um objeto), de uma classe pai (classe que é herdada), por uma classe filho (classe que herda). Tendo tudo isso em mente, o sistema foi criado utilizando a linguagem Java.

2.2. Padrões de projeto

Padrões de projetos são melhores formas de se tomar um determinado problema. Eles são feitos por projetistas que buscam formas de evitar a recriação da roda, determinando formas eficazes de trabalhar com classes e objetos sem ter que criar códigos do zero. Os padrões de projeto escolhidos para o desenvolvimento do sistema foram o MVC e o Facade. O MVC (Model-View-Controller), é dado na forma de separar um sistema em 3 camadas. Respectivamente, view, controlador e modelo. A camada da view é responsável pela interação com o usuário. Dela saem mensagens de requisições do usuário para o sistema. A camada do controller é a responsável por receber essas requisições, e mandar mensagens para o modelo para atualizar seu estado ou mandar mensagens a view alterando a exibição do modelo. E a camada do modelo tem como trabalho armazenar os dados e os validando. Nela constam a parte da modelagem de dados e regras de negócio. O Facade serve como um simplificador de processos. Usado em conjunto com o MVC ele aje entre a view e o controller. Em um sistema que contém subsistemas, o Facade simplifica suas chamadas, dando a view um acesso mais fácil.

2.3. Manipulação de arquivos

De uma variedade de opções de armazenamento físico (não na memória RAM), de dados, foi escolhido arquivos de texto. A escolha foi feita por eles serem mais fáceis de manipular do que um banco de dados por exemplo.

2.4. Ordenação de dados

Ordenar ou classificar alguma coisa é organizá-la de uma maneira definida. Pode ser um conjunto numérico ou dados. Um algoritmo de ordenação é um conjuntos de passos

definidos de forma mais eficiente possível e que ponha o que for pedido em uma determinada ordem. Tomando pessoas como algo a ser ordenado, é possível ordená-las tomando por ordem o nome, a idade, o tamanho, e etc… podendo os deixar em ordem crescente ou decrescente.

2.5. Estruturas de dados

O segredo de muitos dos programas que executam com rapidez e eficiência está do bom uso de estruturas de dados, ou seja, onde e de que forma os dados do programa estão sendo armazenados e acessados. No sistema, é utilizado Array (Vetores) para armazenamento das cartas dentro de um baralho. Listas Encadeadas para armazenar usuários e jogadores, e pilhas que armazenam os baralhos das partidas.

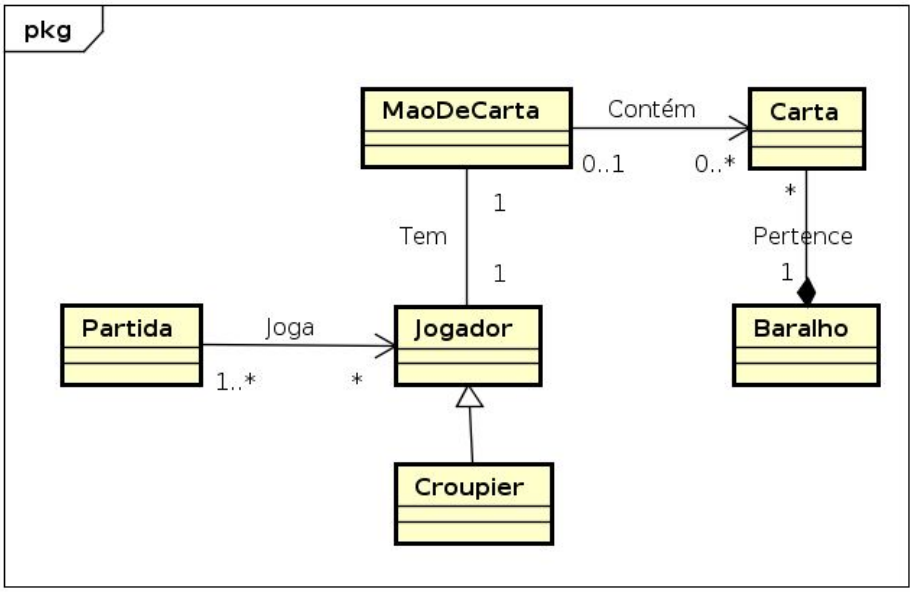
2.6. Ferramentas extras

Para a criação do software foi utilizado Linux Mint 18.3 como sistema operacional, e o ambiente integrado de desenvolvimento Netbeans IDE 8.2.

3. Metodologia

Tomando como pontapé inicial o modelo conceitual contido no descritivo do problema (Figura 1), e os requisitos contidos no descritivo, foram feitos cartões CRC que representavam as classes, as responsabilidades de cada uma e seus contribuidores (classes ajudantes caso uma classe não seja auto suficiente), para um melhor entendimento do problema.

Figura 1. Modelo Conceitual do Sistema



Fonte: Descritivo do Problema

Pegando como base o modelo conceitual e os cartões CRC, foi feito o diagrama de classes do projeto (segue na pasta resources do arquivo comprimido, devido ao tamanho), que foi uma ferramenta de auxílio para a criação dos testes e

consequentemente do código do sistema. O diagrama em si foi uma ferramenta de grande ajuda pois auxiliou a compreensão da abstração do funcionamento do sistema e do relacionamento entre os objetos. Tendo em mãos essas informações, o próximo passo tomado foi a metodologia Test-Driven Development (TDD), que basicamente é escrever testes de funcionamento antes mesmo de escrever o código do produto. Isso torna o código mais confiável e menos suscetível a bugs. Assim como todo o código do sistema, os testes foram escritos em linguagem Java. Os testes verificam se os objetos que serão utilizados no sistema (Jogadores, Croupier e etc…), estão sendo criados corretamente, e se seus métodos funcionam como proposto.

O planejamento que antecede a criação de um sistema como esse é essencial, pois torna seu desenvolvimento mais fácil. Pequenas dúvidas são facilmente eliminadas e é gerada organização.

3.1. Onde entra o MVC

Após essas iniciais etapas de abstração do projeto, foi feita a separação de classes. Cada classe seguiu para sua determinada camada do padrão MVC, Fora o Facade que ficou no seu devido lugar. Foram criados dois controllers: um para a manipulação dos arquivos de texto, e um outro para que em conjunto com a classe partida, conduzisse o decorrer do jogo. Foi buscado seguir a metodologia TDD até onde era possível (por causa do prazo de entrega curto). Então foram criados os testes para as prováveis funcionalidades dos objetos, criadas as classes e os objetos a serem testados. Alterações foram feitas nas classes até que os objetos passassem nos testes.

Na model ficaram os objetos que fazem parte dos dados a serem modelados: carta, baralho, jogador, croupier, mão de carta e partida.

Tendo ficado claro como requisito que o sistema deveria trabalhar com arquivos, foi feito um estudo sobre como a linguagem escolhida para a criação do sistema trabalha na manipula arquivos e quais devem ser os cuidados a serem tomados ao fazê-lo. O controller de arquivos ficou responsável por fazer essa manipulação. No sistema ele está sendo utilizado para conduzir a leitura de dados de usuários cadastrados que estão armazenados em um arquivo de texto que está dentro da pasta do projeto. Esses dados lidos são passados para uma lista encadeada (uma estrutura de dados), de usuários. Essa lista de usuários é utilizada para selecionar quais usuários irão participar das partidas. Este controller também é utilizado para gerar um arquivo de texto de saída mostrando os usuários cadastrados, suas pontuações e a quantidade de partidas ganhas por cada usuário. Esse arquivo de texto é ordenado por maior pontuação, ou seja, o usuário que tiver maior pontuação fica no topo, seguido pelo segundo maior e etc… Essa ordenação é feita pelo algoritmo de ordenação QuickSort.

O controller de partida trabalha junto com a classe partida, ambos realizam todo o processo da partida de blackjack. As ações ocorridas na classe partida são descritas em texto e armazenadas em um histórico na forma de lista encadeada. Esse histórico, é por sua vez exibido pela view.

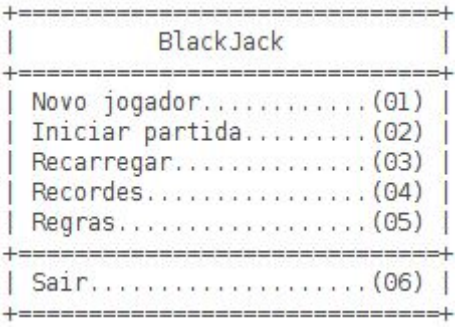
Na view roda o menu principal do jogo. Para cada opção selecionada, a view faz chamadas ao Facade, que por sua vez, chama o controller e os métodos do controller adequados a solicitação. Na view estão os métodos que desenham as bordas do menu

principal, dos menus secundários, as bordas que ficam ao redor da área da partida e o painel que exibe as regras do jogo.

4. Resultados e discussão

O resultado do projeto é um sistema que gerencia partidas no famoso jogo de cartas BlackJack pelo console. É possível jogá-lo simultaneamente com até cinco jogadores. É possível cadastrar um novo jogador sempre que for solicitado, digitando a opção 1 do menu principal (Figura 2).

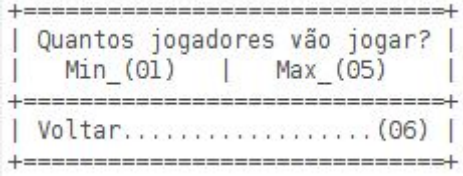
Figura 2. Menu principal do programa



Fonte: Próprio Autor

Cadastrar um novo jogador com login e senha cria um novo item na lista de users. E por consequência, um novo item nos arquivos de texto. Todos os jogadores recebem 100 pontos ao serem cadastrados. Em cada partida, eles automaticamente apostam 10 pontos. Dos jogadores que perderem, são retirados 10 pontos, do que vencerem, são adicionados 10 pontos, dos desistentes é retirado a metade desses pontos e aos jogadores que empatarem nada é retirado. Se a pontuação do usuário for menor do que 10, ele deverá recarregar seus pontos. Escolher a opção 3 do menu principal (Figura 2), faz com que o sistema peça login e senha do usuário que deseja recarregar seus pontos. Escolher a opção 4 faz com que o sistema exiba a lista de jogadores com suas pontuações e partidas ganhas, ordenado por maior pontuação. Escolher a opção 5 faz com que o sistema abra um painel feito em swing que exibe as regras do jogo BlackJack. Escolher a opção 6 faz com que o sistema finalize. As partidas são jogadas quando escolhida a opção 2 do menu. Qualquer letra ou número que não esteja no intervalo disponível não é aceito. Ao ser digitada a opção 2 do menu, são exibidos submenus que caracterizam o tipo de partida desejada. O primeiro desses submenus é o de escolha de quantidade de jogadores (Figura 3), onde é escolhido quantos jogadores participarão da partida.

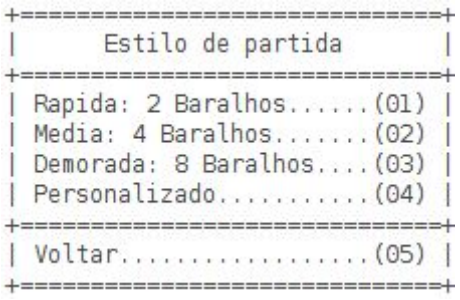
Figura 3. Submenu de escolha de quantidade de jogadores



Fonte: Próprio Autor

O segundo submenu é o de escolha de quantidade de baralhos (Figura 4). submenu, é escolhido com quantos baralhos a partida acontecerá.

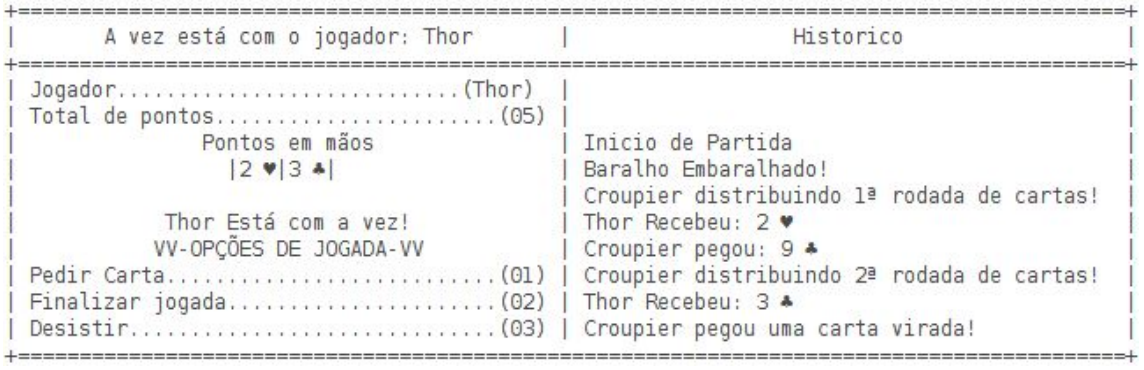
Figura 4. Submenu de escolha de quantidade de jogadores



Fonte: Próprio Autor

Após a quantidade de baralhos ser escolhida, para a quantidade de jogadores escolhida, é obrigatório fazer o login de cada um. Logins feitos, a partida começa (Figura 5), o jogador atual tem a opção de pedir mais uma carta (opção 1), finalizar jogada (opção 2), e desistir da partida (opção 3). É exibido o histórico do lado direito desse submenu, e ele é atualizado a cada jogada.

Figura 5. Submenu de jogadas e visualização do histórico da partida

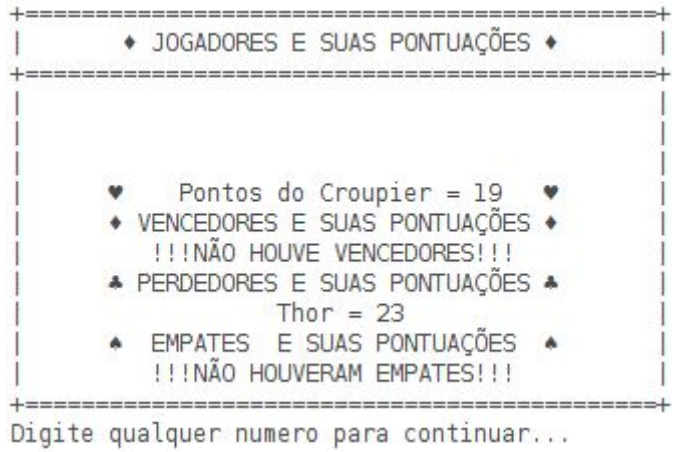


Fonte: Próprio Autor

E depois de todos os jogadores terem jogado seus turnos e croupier também, é feito o balanço da partida. É verificada a pontuação do croupier, e depois comparada com a pontuação de cada um dos outros jogadores.

Quem ganhou, quem perdeu e quem empatou com o croupier (Figura 6), é mostrado em um painel no console.

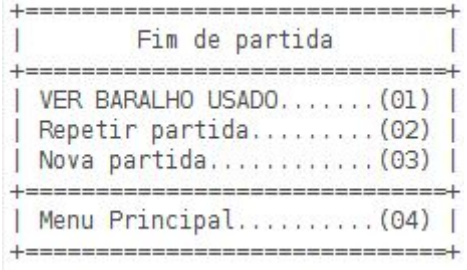
Figura 6. Submenu de status da partida.



Fonte: Próprio Autor

Para finalizar, é mostrado o último submenu (Figura 7), que dá a opção de ver o baralho da última partida na ordem em que ele iria sair e ordenado por naipe e número (opção 1), repetir a partida com os mesmos jogadores (opção 2), jogar uma nova partida (opção 3), e voltar ao menu principal (opção 4).

Figura 7. Submenu mostrado no final de cada partida.



Fonte: Próprio Autor

Qualquer tecla digitada nos submenus que não esteja dentro das teclas permitidas, não serão aceitas e não farão ação alguma no sistema.

5. Conclusão

Depois de todo o processo de desenvolvimento do sistema, o software em questão cumpre todos os requisitos obrigatórios solicitados no escopo do problema. Porém, para fim de otimização, seria interessante utilizar o padrão de projeto Observer no sistema, tendo como o observado a partida, onde a cada atualização, os observadores (view por exemplo), seriam notificados dessa mudança. Aumentar a quantidade de testes unitários seria o ideal, já que o prazo de entrega não proporcionou a criação de muitos, e mudar a interface gráfica atual por uma feita em javaFX ou swing iria trazer uma elegância ao software.