Problema 02: Jogo de cartas Blackjack

Uellington Da Conceição Damasceno

[1]Curso de Engenharia de computação - Universidade Estadual de Feira de Santana (UEFS)
Av.Trasnordestina, s/n - Novo Horizonte, Feira de Santana - BA, 44036-900

Uellington99@gmail.com

Resumo. *Este artigo trata de um relatório do desenvolvimento de um software (que faz algo). Nele é detalhado os requisitos do sistema, os recursos de programação utilizados, o processo que levou até a solução do problema, como também o desenvolvimento do programa. A linguagem de programação POO utilizada foi o Java, e o processo de desenvolvimento foi iterativo e incremental em dupla.*

1. Introdução

Hoje é muito perceptivo a crescente evolução das tecnologias que compõem nosso cotidiano, as quais as vezes se tornam quase imperceptíveis, e que estão sempre substituído "coisas" e "meios". Também é muito claro como a tecnologia influência nos jogos, seja com a criação de novos jogos, quanto na reformulação de jogos antigos, com o intuito de melhorar dissemina-los.

Assim, durante uma viagem a cidade de Las Vegas, nos Estados unidos da América, Mauricinho Rico ficou muito encantado com o jogo *BlackJack*, o qual era jogado em todos os cassinos da cidade. Após a viagem, Mauricinho Rico, sabendo da capacidade da Empresa *Jr* em desenvolver softwares solicitou que a empresa reformulasse o jogo *BlackJack* em forma de *software*.

O jogo tem como funcionalidade a possibilidade de cadastrar jogadores, onde esses jogadores teriam um nome de usuário e senha, a possibilidade de após a rodada verificar o placar que contém o nome do usuário com sua respectiva pontuação e quantidade de partias vencidas e também a possibilidade de verificar o baralho utilizado na última partida jogada.

Neste relatório, será apresentado como foi feito e quais critérios foram considerados para a construção da base do projeto. Para um melhor entendimento de como o *software* foi desenvolvido este relatório foi dividido em quatro partes, sendo elas:Introdução Metodologia, resultados e discussões e por fim conclusão.

2. Metodologia

Com a problemática em pauta, foram discutidos e analisados os requisitos e restrições do programa nas sessões tutoriais e com base nisso algumas questões foram levantadas como por exemplo as que se dizem respeito a: Manipulação de arquivos, estrutura de dados e ordenação de dados.

2.1. Manipulação de arquivos

A manipulação de arquivos partiu da necessidade de armazenar um jogador e suas respectivas informações, de modo que pudessem ser utilizadas mesmo com a finalização do programa.

Para cumprir esse requisito foi desenvolvido um controlador de arquivos que possui um referencia a dois arquivos (Padrões e que só podem ser alterados por linha de código) sendo elas: “Logins.bin”, responsável por armazenar o usuário, senha, pontos e quantidade de partidas vencidas de um determinado jogador e “Pontuação.txt”que por sua vez armazena a informações semelhantes ao “Logins.bin”, diferenciando somente no fato de não possui a senha do usuário.

Esses arquivos são sincronizados são atualizados sempre que ocorre as seguintes ações:

- Cadastro de novo jogador,
- Recarrega de pontos, e
- Final de partida.

o processo de atualização consiste em apagar os registros que o arquivo possui e reescrevelos já com as atualizações.

A escolha de atualizar o arquivos arquivos em pontos distintos e não deixar para fazer isso em um só ponto se deu por causa das limitações que está associada a finalização do programa sem que o ponto “Checkpoint”chegasse.

Durante a manipulação dos arquivos foi necessário compreender o funcionamento das exceções que são lançadas pelos métodos que abre, fecha, direcionam o fluxo de informações e consequentemente foi necessário entender o funcionamento das estruturas (try catch e throws) responsáveis por auxiliar no tratamento dessas exceções.

2.2. Estrutura de dados

Durante a analise do modelo conceitual notou-se que haveria a necessidade de utilizar estrutura de dados. tendo em vista o grande “leque”de possibilidades e a obrigatoriedade na utilização da Pilha em algum momento a distribuição de as estruturas ficaram da seguinte maneira:

- Array - Para criação do baralho e ordenação do mesmo,
- Lista Encadeada - Para lista de jogadores, e
- Pilha - Para estado do baralho durante partida;

Para embaralhar o baralho foi pensado em duas possíveis maneiras, sendo elas: tirar da pilha e colocando em posições aleatórias de um array ou percorrer o array trocando a carta atual com uma carta de posição aleatória.

A primeira solução surgiu por causa do funcionamento LIFO (*Last in, Frist out*) da pilha que só permite tirar a carta do topo. Então para que houvesse o embaralhamento seria necessário desempilhar as cartas colocando em posições aleatórias de um array e por fim empilha-las novamente.

A segunda solução mostrou ser mais simples. pois, somente seria necessário percorrer o a coleção trocando a carta atual pela carta que está em uma posição aleatória e por fim empilhar as cartar para que o baralho pudesse ser utilizado em partida.

Pelos motivos justificados a cima, criar o baralho inserindo as cartas em um array se mostrou consideravelmente menos desgastante se comparado com a inserção direto na pilha. Já a lista encadeada e a pilha foram utilizadas por escolha de projeto e obrigatoriedade respectivamente já o uso do array.

2.3. Ordenação de dados

Dado a necessidade de exibir as cartas restantes de um baralho após a partida e a lista de jogadores ordenadas por um determinado critério sendo eles: nipe e face para as cartas e maior pontuação para os jogadores. notou-se que era necessário implementar um algorítimo de ordenação.

Dentre todos os algorítimo de ordenação três se destacaram sendo eles: Bucket Sort, Insertion Sort e Quick Sort.

O insertion sort apesar de não se tão eficiente (Melhor caso O(n2)) quanto os concorrentes entrou na lista pela logica utilizada se bastante próxima com a que normalmente é usada para a ordenação de cartas de um baralho.

O bucket sort por sua vez mostrou-se bastante eficiente (Melhor caso O(n+k)) e um excelente concorrente entretanto por motivos escolha de projeto o quick sort foi o escolhido como método de ordenação.

O quick sort é um algoritmo de ordenação baseado em comparações, que tem como complexabilidade O(n2) no pior caso e O(n log n) no melhor caso. Ele adota a estratégia de divisão e conquista, essa estratégia consiste em rearranjar as chaves de modo que as chaves que possuem o menor valor precedam as chaves com valores maiores.

Para que o quick sort fosse utilizado por múltiplas classes foi necessário o uso a classe abstrata *Comparable* a qual permite que o classe que implementou trone-se comparável através de um determinado critério.

Vale também ressaltar que o padrão de arquitetura de software utilizado foi o MVC (Model, View, Controller), que contou com dois controllers (ControllerArquivos e ControllerPartida) sendo intermediado por um Facade. Foram realizados testes de unidades através do JUnit e a plataforma de desenvolvimento utilizada foi o NetBeans IDE 8.2 com a linguagem JAVA.

3. Resultados e discussões

Assim que o software é inciado ele verifica se o arquivo padrão de usuários existe, caso o arquivo seja encontrado ele irá ler e carregar as informações dos usuários, caso contrario ele irá criar um novo arquivo padrão. Em seguida o usuário terá acesso a um menu com 6 opções sendo elas:

- Cadastrar novo jogador;
- Iniciar partida;
- Recarregar;
- Recordes;
- Regras;
- sair;

Nesse momento o software vai está aguardando um valor inteiro entre 1 e 6, caso o usuário digite um valor fora do intervalo definido o método responsável pela leitura (lerInt) das inputs do usuário irá notifica-lo pedindo para que digite um valor dentro do intervalo. bem como notificará caso o valor digitado seja uma letra ou palavra.

3.1. Cadastrar novo jogador

A opção cadastrar novo jogador é auto explicativa. pois seu objetivo é receber o nome do usuário e a senha e em seguida verificar se ele pode ou não ser cadastrado.

3.2. Iniciar partida

Ao selecionar a opção iniciar partida o usuário será direcionado para um menu que irá questionar quantos jogadores irão participar da partida. (levando em consideração que o mínimo é 1 jogador o máximo de 5) caso o numero de jogadores escolhido seja menor que a quantidade de jogadores ou não tenha nenhum jogador cadastrado, irá aparecer uma mensagem perguntando se ele não deseja cadastra um novo jogador.

Caso exista jogadores suficientes para a partida o usuário será novamente direcionado para um menu que define a quantidade de baralhos que a partida deve contar. Nesse menu é possível jogar com: 2, 4, 8 ou com um numero personalizado de baralhos.

Logo após selecionar o numero de baralhos o(os) jogador(ores) deverão fazer login utilizando o user e a senha previamente cadastrados no sistema. caso o usuário inserido possa uma pontuação menor que 10 aparecerá uma mensagem informando que ele não possua pontos suficiente e o permitirá "comprar"pontos e assim que ele estiver com uma pontuação suficiente poderá participar normalmente da partida.

Finalizados os logins e todos os jogadores devidamente inseridos na partida, começa a rodada de distribuição de cartas e a partir dai ambos os jogadores irão visualizar a interface de partida e podaram escolher entre: pedir carta, finalizar jogada e desistir. A partir desse pontos o software foi desenvolvido para possuir uma dinâmica muito próxima a do jobo BlackJack.

Quando todos os jogadores finalizam a jogada, desistem ou perdem começa a premiação. Nesse momento será feito todas as atribuições de pontos e partidas vencidas ao jogadores, com base na comparação dos pontos em mãos com os pontos em mãos do croupier.

Após finalizar a toda dinâmica da partida o usuário será direcionado a um menu que permitirá visualizar o baralho na ordem de saída e ordenado, também poderá repetir a partida com os mesmo jogadores, iniciar uma nova partida (Escolhendo novos jogadores) ou voltar para o menu principal.

3.3. Recarregar

Durante a analise do problema notou-se que existe um sistema de pontuação. porém não havia nenhuma especificação de como deveria ser desenvolvido. por esse motivo surgiram varias possíveis maneiras de desenvolver a pontuação. mas, a escolhida para esse *software* foi o sistema de recarrega.

O sistema de pontuação por recarrega consiste basicamente na possibilidade de permitir ao usuário poder "comprar"pontos. Inicialmente todo jogador recém cadastrado no sistema começa com 100 pontos, para entrar na partida ele precisa de no mínimo 10 pontos. pois, ao final da partida será contabilizado e ele ganhará 10 pontos caso vencer, perderá 10 caso desista, perderá 5 caso perca do croupier e não ganhará e nem perderá caso empate com o croupier.

Caso o usuário não possua pontos suficiente ele pode fazer login no menu de recarga ou antes do inicio da partida e poderá “comprar”: 25, 50, 75 e 100 pontos. após efetuar a recarrega o usuário está apto para poder jogar imediatamente caso queira.

4. Conclusão

Após todo o trabalho de desenvolvimento, o programa chegou a um ponto do qual consegue cumprir todos os requisitos mínimos necessários. Requisitos esses que variam desde a leitura e a escrita dos arquivos, mecânica da partida com resultados, ordenação das informações até a interação final com o usuário.

Na atual versão do programa não foi encontrado nenhum tipo de falha com relação ao que se diz respeito a interação com o usuário. Entretanto, é possível que exista algum problema não descoberto, que pode ser desencadeado depois de algumas sequência de entradas ou manipulação indevida do arquivo de leitura padrão utilizado para armazenar as informações os jogadores.

Para um melhor desempenho do programa, poderia ser implementado mais testes unidade para verificar a integridade do programa já existe, poderia também ocorrer uma refatoração no código para ampliar a capacidade de modularização, um melhor tratamento com relação as informações geradas pela manipulação dos arquivos e melhorias na interface substituindo-a por uma interface gráfica utilizando alguma ferramenta como javaFX ou swing.