Especificação de Requisitos do Software Paciência Spider

Autores:

Felipe Zarattini Miranda, Felipe Holanda Bezerra, Gabriel Maia Guzowski

Versão: 1.0.0

Data:09/04/2015

1- Introdução

1.1- Objetivo do documento:

Este documento visa descrever de forma completa como deverão ser as características e o funcionamento da aplicação Paciência Spider.

2- Características da aplicação

2.1- Objetivo:

A aplicação é uma variação do popular jogo de cartas Paciência, porém com suas próprias regras e características individuais. O objetivo do jogo é reorganizar as sequências mostradas na tela seguindo regras específicas de modo a completar sequências de A até K de mesmo naipes até que o baralho esteja completamente organizado.

2.2- Funcionamento:

- Ao iniciar a aplicação, abrirá uma tela do prompt de comando do Windows com um pequeno texto explicando a simbologia do jogo.

- O jogador será perguntado quanto a que dificuldade deseja jogar.

- As 10 sequências de cartas serão então mostradas na tela para que o jogo possa começar.

- Cada carta da sequência será representada por um valor (A, 2, 3, ... , J, Q, K) seguido do símbolo de seu naipe, ou de símbolos específicos quanto a estar virada para baixo ou bloqueada.
- Serão mostradas na tela as opções do jogador: mover cartas (M), pedir cartas de um dos cinco montes (P) ou salvar e sair do jogo (S).

- Sempre que houver alguma ação por parte do jogador, as sequências serão atualizadas e mostradas na tela novamente. Caso a ação seja salvar e sair, os dados do jogo serão salvos e o programa terminado, possibilitando que se continue de onde parou em uma nova sessão.

2.3- Simbologia:

A aplicação deverá utilizar dos seguintes símbolos:

- ♣ – Naipe Paus
- ♥ – Naipe Copas
- ♦ – Naipe Ouros
- ♠ – Naipe Espadas
- X – Carta virada para baixo
- │A♦│- Carta bloqueada (no exemplo, Ás de Ouros)

2.4- Comandos do Jogo:

A aplicação deverá possuir 3 comandos principais, além das escolhas de valor, naipe, sequência fonte e destino no caso de uma movimentação.

- **M**: Mover cartas de uma sequência para outra. Poderão ser movidas cartas individuais, caso estejam no fim de uma sequência, ou grupo de cartas, caso seja escolhida uma que esteja no meio de uma sequência. O jogador será questionado quanto ao valor da carta que deseja mover, a letra inicial de seu naipe, a sequência onde se encontra e a sequência para onde deseja que seja movida.

- **P**: Utiliza de um dos cinco montes principais para pedir cartas para o jogo. Esse comando irá adicionar uma carta a cada uma das sequências do jogo. Se ao ser adicionada, a carta não for continuação da sequência já existente, as cartas acima serão todas bloqueadas.

- **S**: É utilizado para salvar o estado do jogo e finalizar a sessão.

3- Regras do Jogo

- Um grupo de cartas só poderá ser movido se todas as cartas que o compõem forem do mesmo naipe.

- Uma carta, ou grupo de cartas só poderá ser movida para uma sequência, caso a última carta da sequência de destino seja de um valor acima.

- O jogador pode mover uma carta ou grupo de cartas para uma sequência de naipe diferente, porém as cartas que já existiam na sequência destino serão bloqueadas e não poderão ser manipuladas.

- Não será possível mover cartas bloqueadas ou viradas para baixo.

- O jogador poderá pedir cartas apenas cinco vezes.

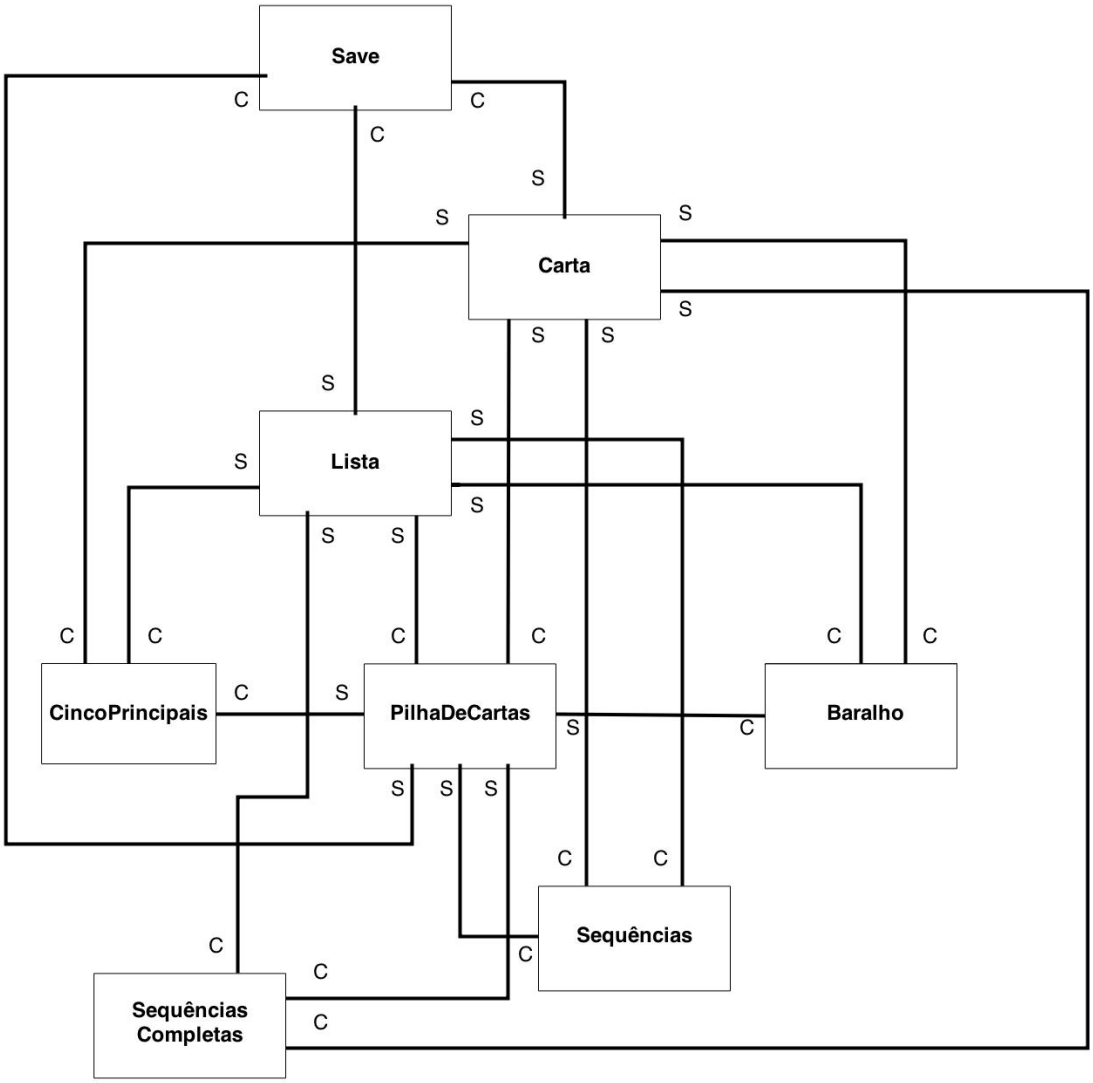
- Não será possível pedir cartas dos cinco montes caso haja uma sequência vazia em jogo, ou seja, sem cartas.

- Será possível mover qualquer carta para espaços de sequências vazios na mesa.

- Quando completar uma sequência (de A até K), essa sequência será removida para um espaço separado do jogo.

- O objetivo do jogo será completar 8 sequências (totalizando as 104 cartas em jogo), todas de mesmo naipe.

Arquitetura do Programa

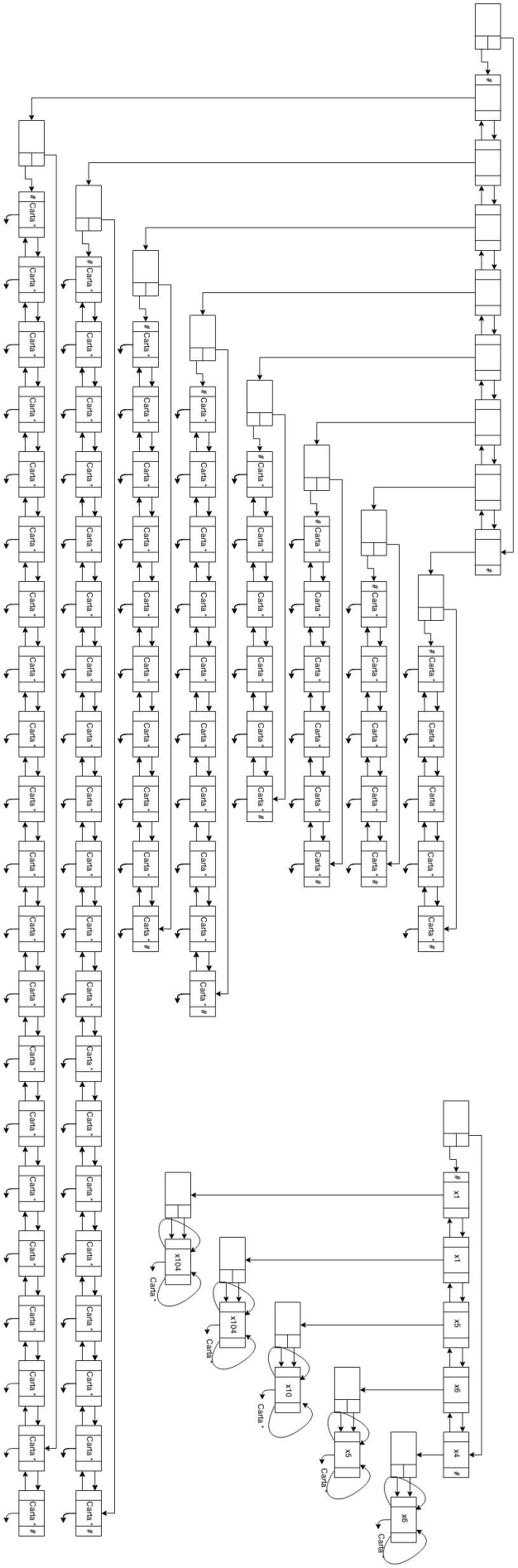


Principal

OBS.: Módulo Principal é cliente de todos os módulos

5. Estrutura Principal

(Modelo e Exemplo)



Assertivas Estruturais:

1. Um nó da lista deve necessariamente apontar para uma carta ou para uma cabeça de uma outra lista.

   ```
   No->pElem == *Carta || No->pElem == *tpCabeca
   ```

2. Assertiva estrutural da lista duplamente encadeada genérica.

   ```
   Se no->pAnt != NULL, então no->pAnt->pProx == no
   ```

   ```
   Se no->pAnt != NULL, então no->pAnt->pProx == no
   ```