Relatório Projeto Paciência
Igor Garcia

Explicação geral de como o projeto implementa o jogo paciência:

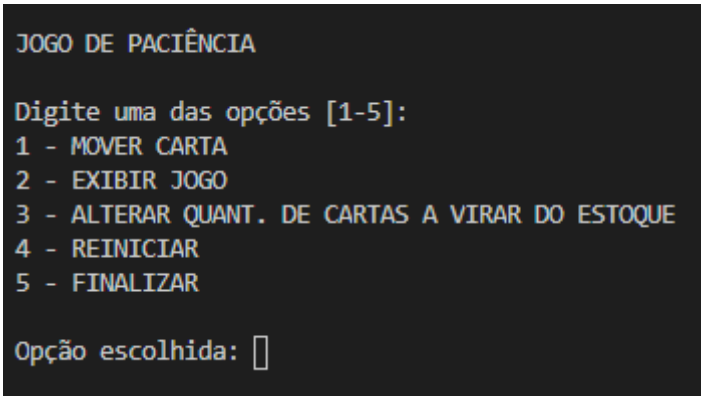
O projeto segue o padrão de design MVC, onde o pacote View possui a classe "App", que faz a interação com o usuário pelo terminal, chamando alguns dos principais métodos definidos dentro da classe Controller, como o moverCarta(), exibeJogo(), reiniciar() e finalizaJogo().

Dentro do pacote Model estão as classes "Carta" e "Baralho". A classe "Carta" possui os atributos de uma carta de baralho e métodos para obter ou setar valores nesses atributos. A classe "Baralho" inicializa as pilhas que serão usadas no jogo e define métodos para obter valores delas ou setar valores nelas. Além disso, esta classe é responsável também por inicializar uma pilha principal de baralho que será usada durante o jogo.

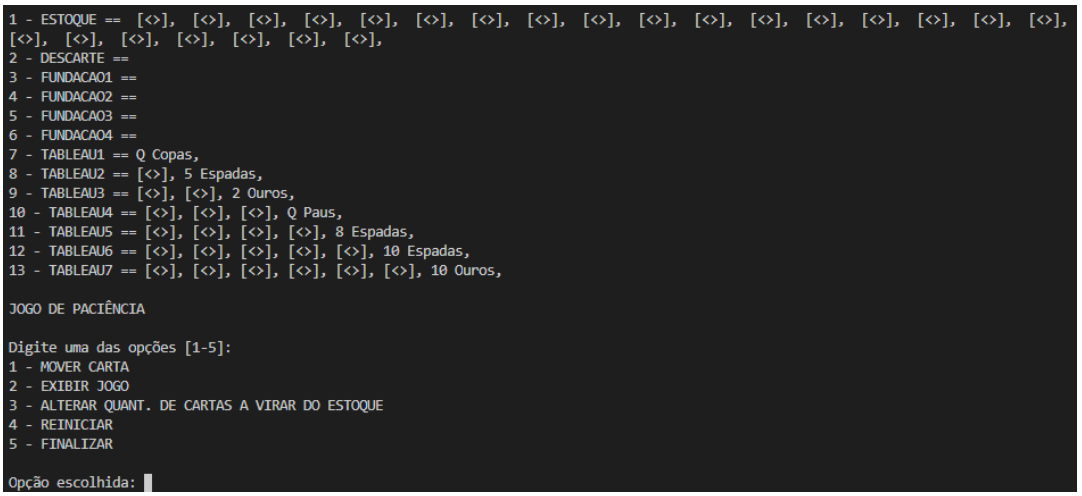
A classe "Controlador", no pacote Controller, é a mais robusta. Ela possui métodos para fazer todas as movimentações de cartas do jogo e todas as checagens para validar se um movimento pode ou não ser feito pelo jogador. Além disso, "Controlador" também exibe todo o jogo no terminal, finaliza ele e o reinicia.

Explicação de como fazer para jogar:

Todo o jogo possui menus como o mostrado abaixo, onde os números digitados definem a ação a ser realizada:

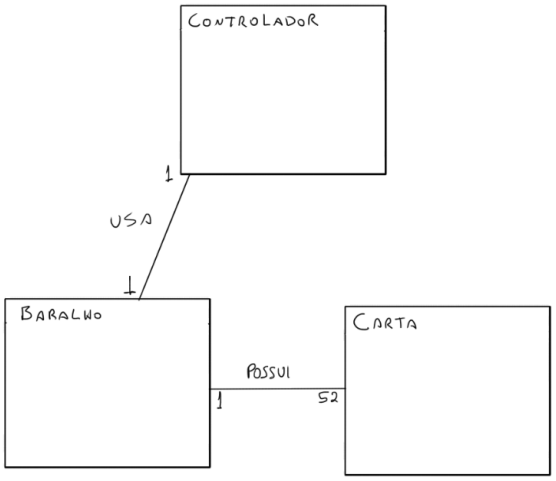


A opção de "EXIBIR JOGO", por exemplo, tem esta tela abaixo como saída:



Fora isso, não há nenhum outro detalhe que fuja desse padrão. Todas as ações que podem ser feitas no jogo são bem explicadas nos menus no terminal, não havendo momento de dúvida para o jogador.

Diagrama UML simples:

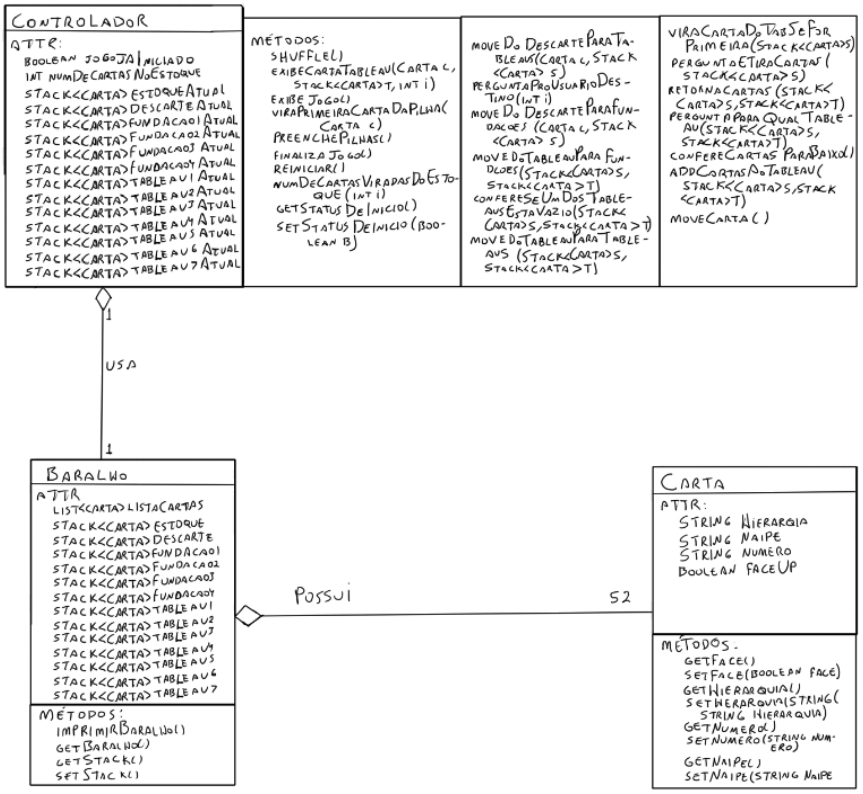


O controlador gerencia todo o jogo. Ele instancia apenas uma pilha de baralho e manipula ela durante a partida.

A entidade do baralho inicializa pilhas do tipo Carta e cria um array para as 52 cartas do baralho.

A entidade Carta define as variáveis do objeto Carta e seus respectivos getters e setters.

Diagrama UML com detalhes de métodos e atributos:



Padrões utilizados:

A ideia inicial foi de usar o padrão Singleton, tendo em vista que uma partida deveria ter apenas uma instância de baralho. Sendo sincero, não tenho certeza absoluta se, tecnicamente, o padrão foi aplicado como é dado em livros, mas ao longo da partida, há apenas um baralho e as cartas não são repetidas, são sempre as mesmas 52, logo, a teoria de como o Singleton funciona foi bastante útil no processo.

Quanto a padrões que possam vir a ser utilizados, faz sentido cogitar o uso do observer, já que há no programa a dinâmica de uma entidade tomar uma ação baseado na mudança de outra, ex: Se houver uma contagem de pontos, a entidade encarregada disso poderia ser notificada por outra quando o jogo tivesse acabado, ou que uma fundação foi acrescida de uma carta, para que ela incrementasse pontos.

Pontos fortes e fracos do projeto:

Todas as funções que não envolvem a movimentação de apenas uma carta por vez entre os tableaus foram testadas à exaustão, logo, creio que os pontos fortes englobam a maior parte do programa. Os pontos fracos estão, por eliminação, nas funções responsáveis por mover mais de uma carta por vez entre os tableaus, verificar se a movimentação é válida ou não e retorná-las, caso não seja. Essa parte do programa foi a única realmente complicada e a última a ser implementada. Ela aparentemente funciona corretamente, porém é a mais passível de falhas.