PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Algoritmos e Estruturas de Dados – 2/2024

TRABALHO PRÁTICO

Prof. Edwaldo Soares Rodrigues

Instruções:

I. O trabalho deverá ser feito individualmente.
II. O trabalho deverá ser realizado usando a linguagem de programação C#.
III. Deverão usar os conceitos aprendidos na disciplina Algoritmos e Estruturas de Dados, levando em consideração as melhores estruturas para representar os itens do jogo (monte de cartas, área de descarte, as mãos dos jogadores e afins). **OBS: É solicitado que seja utilizada ao menos uma estrutura de dados linear (implementação manual, não Collections), ao menos uma estrutura de dados flexível (implementação manual, não Collections) e ao menos uma estrutura de dados via Collections.**
IV. O trabalho deverá ser entregue até a data 10/12/2024, via Canvas.
V. Somente poderão ser utilizadas estruturas e conceitos vistos em sala de aula.
VI. A avaliação do trabalho será por meio de apresentação do código.
VII. Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;

Problema – Rouba Montes:

Um dos jogos de cartas mais divertidos para crianças, pela simplicidade, é o Rouba- Monte. O jogo utiliza um ou mais baralhos normais (52 cartas, de às à rei, em cada um dos 4 naipes) e tem regras muito simples. Cartas são distinguidas apenas pelo valor (ás, dois, três, . . .), ou seja, os naipes das cartas não são considerados (por exemplo, ás de paus e ás de ouro têm o mesmo valor).

Inicialmente as cartas são embaralhadas e colocadas em monte na mesa de jogo, chamado de monte de compra, com face voltada para baixo. Durante o jogo, cada jogador mantém um monte de cartas, com face voltada para cima; em um dado momento o monte de um jogador pode conter zero ou mais cartas. No início do jogo, todos os montes dos jogadores têm zero cartas. Ao lado do monte de compras encontra-se uma área denominada de área de descarte, inicialmente vazia, e todas as cartas colocadas na área de descarte são colocadas lado a lado com a face para cima (ou seja, não ficam uma sobre a outra).

Os jogadores, dispostos em um círculo ao redor da mesa de jogo, jogam em sequência, em sentido horário. As jogadas prosseguem da seguinte forma:

- O jogador que tem a vez de jogar retira a carta de cima do monte de compras e a mostra aos outros jogadores; vamos chamar essa carta de carta da vez.

- Se a carta da vez for igual à carta do topo de um monte de um outro jogador, o jogador "rouba" esse monte, colocando-o em seu próprio monte, coloca a carta da vez no topo do seu monte, face para cima, e continua a jogada (ou seja, retira outra carta do monte de compras e repete o processo). Duas cartas são consideradas iguais se tiverem o mesmo valor. Caso a carta da vez seja igual ao topo de dois ou mais montes, deve-se roubar apenas o maior monte (monte com mais cartas). Se houver empate em relação ao tamanho dos montes, deve-se escolher aleatoriamente um dos montes para roubar.

- Se o teste acima falhar, o jogador verifica se a carta da vez é igual a alguma carta presente na área de descarte. Caso seja, o jogador retira essa carta da área de descarte colocando-a no seu monte, juntamente com a carta da vez no topo, com as faces voltadas para cima, e continua a jogada (ou seja, retira outra carta do monte de compras e repete o processo).

- Se o teste acima falhar, o jogador verifica se a carta da vez é igual a carta do topo de seu próprio monte. Caso seja, o jogador coloca a carta da vez no topo de seu próprio monte, com a face para cima, e continua a jogada (ou seja, retira outra carta do monte de compras e repete o processo).

- Se a carta da vez for diferente das cartas da área de descarte e das cartas nos topos dos montes, o jogador a coloca na área de descarte, com a face para cima, e a jogada se encerra (ou seja, o próximo jogador efetua a sua jogada). Note que esse é o único caso em que o jogador não continua a jogada.

A partida termina quando não há mais cartas no monte de compras. O jogador que tiver o maior monte (o monte contendo o maior número de cartas) ganha a partida. Se houver empate, todos os jogadores com o monte contendo o maior número de cartas ganham a partida. Ao final de cada partida, deve-se perguntar se o usuário deseja iniciar uma nova partida.

Metodologia:

Deve ser incluída uma classe para representar uma Carta, esta deve ter dois atributos, **número** e **naipe**. Observe que o atributo naipe não será utilizado nesta versão do jogo, mas deve estar presente na classe. Os montes e a área de descarte devem ser compostos por Cartas. Considere que as cartas dama,

valete e rei correspondem, respectivamente, aos valores 11, 12 e 13.

Da mesma forma, deve ser incluída uma classe para representar cada Jogador, contendo os atributos: **nome**, **posição** (posição do jogador na última partida), **quantidade de cartas no monte** (quantidade de cartas na última partida), e uma **fila** contendo o ranking do jogador nas últimas 5 partidas.

Seu programa deve permitir que se escolha a quantidade de jogadores que vão jogar, bem como a quantidade de baralhos que poderão ser usadas no jogo. Em seguida, após a execução de cada partida do jogo, deve ser apresentado o(s) ganhador(es) (nome do jogador, posição e quantidade de cartas no monte). Além disso, deve ser exibido o ranking com os nomes dos jogadores, ordenado pela quantidade de cartas no monte de cada jogador.

Ao término do jogo, o programa deve permitir que o usuário insira o nome de um jogador para visualizar seu histórico de posições nas últimas 5 partidas, no máximo.

O programa deve gerar um arquivo texto com o log das ações executadas em cada partida. Exemplo: "O baralho foi criado com X cartas. Jogadores da partida: [nomes dos jogadores]. Em seguida, indique qual jogador iniciará. Após isso, registre a carta retirada por cada jogador do monte, e continue registrando todas as ações subsequentes ao longo da partida, detalhando cada evento ocorrido."

Observação: quaisquer partes não explicitamente descritas nesta especificação podem ser implementadas a critério próprio.

Critérios de avaliação:

- Escolha das estruturas de dados para representação dos itens do jogo;
- Código legível e bem identado;
- Variáveis, classes e métodos com nomes representativos;
- Uso adequado dos recursos de linguagem;
- Modularização e organização do código;
- Uso de arquivos para registrar os logs da partida;
- Contrário as boas práticas de programação, o código a ser entregue não deve conter nenhum comentário.