Trabalho de Estrutura de Dados

Michel Ribeiro Corrêa

Classe Peca

A classe Peca, contém os 2 números da peça de dominó, no programa, irão existir 28 peças. A classe tem 2 atributos do tipo inteiro, o n1 e o n2.

```
public class Peca {
    public int n1;
    public int n2;
}
```

Classe Node

A classe Node é o nó da lista, por ser uma lista duplamente encadeada, possui 3 atributos: a informação do nó (Peca peca, a peça atribuída ao nó), dois atributos do tipo Node, que contém o próximo nó, e outro que contém o nó anterior.

```
public class Node {
    Peca peca;
    Node prev;
    Node next;
}
```

Classe List

A classe List é a classe que comporta a lista.
Possui 5 atributos, 2 deles serão utilizados somente na lista das peças jogadas.

```
public class List {
    public Node head;
    public Node last;
    public int size;
    public int p1;
    public int p2;
}
```

O atributo head do tipo Node, é o nó da lista que corresponde ao primeiro elemento, em posição.
O atributo last do tipo Node, é o nó que corresponde ao último elemento da lista, em posição.
Em ambos os casos, quando um objeto é excluído sendo head ou last, ele é substituído pelo seu posterior ou antecessor.

No trabalho, foram criadas 3 listas, a lista de peças jogadas, que começa vazia e vai recebendo as peças que os jogadores escolhem jogar, as outras 2 listas, correspondem às peças de ambos jogadores, o player e o bot, distribuídas aleatoriamente de uma outra lista com as 28 peças de dominó.

O atributo size, do tipo inteiro, é o atributo que corresponde ao tamanho da lista, sempre que ocorre uma inserção, ele recebe +1, e quando ocorre uma exclusão de peça, ele recebe -1. Ele é usado diversas vezes, para verificar se a lista está vazia, ou até para percorrer a lista.

Os outros dois atributos (p1 e p2), só são utilizados na lista de peças jogadas. Num jogo de dominó existem 2 “pontas”, 2 possíveis jogadas, e estes atributos realizam esse papel. Quando a primeira peça é jogada pelo bot, o valor do n1 da peça é atribuído ao p1, e o n2 ao p2.
No jogo da imagem abaixo, a variável p1 teria o valor 4, e a variável p2 teria o valor 5.
A próxima peça jogada, obrigatoriamente, teria um desses valores.

Métodos da Classe List

A classe List possui vários métodos.

O primeiro método é o `insertHead(Peca peca)`, esse método recebe o valor de uma peça como parâmetro e realiza a inserção na primeira posição da lista. Se a lista estiver vazia, essa peça também se torna o node last.

O método `insertLast(Peca peca)`, recebe o valor da peça como parâmetro e realiza a inserção da peça na última posição da lista.

O 2 métodos last e head, são os 2 mais utilizados no programa, pois a peça é sempre jogada nos extremos, ou a peça é jogada no Head, ou a peça é jogada no Last, correspondentes às 2 pontas do jogo de dominó.
Métodos de remoção:
O método `removePeca(int i)` recebe um valor inteiro, que corresponde a posição da peça que deverá ser removida da lista. Caso esse valor seja 0, essa função chama o método `removeHead()`, que fará a remoção do primeiro objeto da

lista, o head. Caso esse valor seja igual ao tamanho da lista + 1, o método chamará a `removeLast()`, que removerá a última peça da lista, ou last. Caso não seja nenhuma das opções, um for será rodado, até que chegue na posição do valor i, que removerá o nó atribuído nessa posição.

Esses métodos de remoção são usados constantemente no código, toda vez que uma peça é jogada, essa peça é removida da lista de peças do jogador e adicionada na lista de peças jogadas.

O método `returnNode(int i)` é uma função que retorna o nó correspondente, dependendo da posição informada no parâmetro. Foi usado quando era necessário utilizar os atributos do nó, seja as informações da peça para realizar comparações, ou "varrer" a lista em busca de uma peça.

O último método é o `imprimirLista()` que faz uma função parecida ao toString, porém criei esse método, para colocar várias quebras de linha. A ação dele é como o próprio nome diz.

Main

O programa começa com as 28 peças sendo criadas e armazenadas em uma ArrayList com o nome "pedras". Depois as peças são distribuídas aleatoriamente entre as 2 listas duplamente encadeadas, a lista de peças do Player e a lista de peças do Bot. As funções que realizam essas tarefas são: `criarPedras(); distribuirPecasBot(); distribuirPecasP();`

O jogo começa quando entra no loop `while(true)`. A primeira jogada é feita pelo Bot, que joga uma peça aleatória da sua mão, usando a variável random.
Depois as jogadas são intercaladas, no console aparece todas as peças jogadas e o número das 2 "pontas" do jogo. Depois aparecem todas as peças da mão do player e ele seleciona por meio de um número do teclado, a peça que deseja jogar. Não há verificação no caso do player se essa peça é válida ou não, então cabe ao jogador selecionar a peça adequada.
As funções que realizam as jogadas são: `playerPlay()` e `botPlay()`, ambas funções realizam a jogada, adicionando a peça selecionada a lista de peças jogadas e também realizam a exclusão da peça jogada, também definem a possível nova "ponta" do jogo. Caso seja jogada na ponta 1, realizam a inserção por meio do `insertHead()`, caso seja na outra ponta, por `insertLast()`.
Também existe o caso do Player querer passar sua vez, ou não tiver nenhuma jogada possível, selecionando no teclado o valor -1, o player passa a vez.
Para o computador, ele só passará a vez quando não tiver nenhuma jogada possível. O console sempre informará a jogada do bot, mesmo quando o mesmo passar a vez.

O jogo sai do loop while por meio de um break.
Existem algumas formas para o fim do jogo.
A primeira delas é quando uma das listas de peças dos jogadores fica vazia. Assim, o jogo termina, dando vitória para quem já jogou todas suas peças, antes do player adversário. O outro método de fim de jogo é quando o jogo atinge um estado inconsistente.

Estado Inconsistente

O programa possui uma variável (int travado) que realiza a contagem de quantas vezes uma jogada foi passada, seja pelo bot ou player, indicando que não foram realizadas jogadas. Quando essa variável chega no número 6, o jogo termina, e o jogador com o menor número de peças em sua mão ganha. No caso de ambos jogadores possuírem a mesma quantidade de pedras na mão, o jogo termina em empate. Escolhi o número 6 porque é um número que permite bastante jogadas, e quando não é possível mais jogar nenhuma peça, as jogadas são passadas e não demora tanto para terminar.