UNIVERSIDADE DO VALE DO SAPUCAÍ

SABRINA MEIRELES COELHO TEODORO

TRABALHO DE ESTRUTURA DE DADOS

POUSO ALEGRE

2022

Classe Lista

A classe Lista tem o intuito de representar a estrutura de dados lista duplamente encadeada e é formada por dois atributos da classe “No” de nomes “inicio” e “fim”; uma variável global do tipo inteiro “indice” para indexar os nós na lista; um construtor que não recebe parâmetro, mas quando instanciado o objeto este seta o índice para 0 e pelos métodos:

- Inserir: Recebe um objeto do tipo Entidade para instanciar um nó e atribuir um índice a ele conforme sua posição na lista. Para inserção é utilizado o método “estaVazia” da própria classe que verifica se a lista esta vazia ou populada verificando o atributo “inicio”. Caso este tenha o valor “null”, o método retorna verdadeiro indicando que a lista está vazia e o nó a ser inserido será o inicio e fim da lista. Caso retorne valor false, o nó a ser inserido é adicionado ao final da lista. A cada execução desse método a variável índice é incrementada;
- InserirPorPosição: Diferente do método inserir, este método recebe por parâmetro um valor inteiro que representa a posição do objeto a ser inserido na lista e um objeto do tipo entidade para instanciar o nó. Este método faz uso de outro método da própria classe “buscarPorPosição” que retorna o nó correspondente ao índice buscado ou null, caso inexistente. O objeto retornado é usado para acessarmos o seu anterior e próximo e assim fazer as conexões necessárias para inserção do no na posição. Ao final, o método faz uso de outro método da própria classe “atualizaIndice” que faz a contagem dos nós atualizando-os de seu novo índice após uma inserção.
- BuscarPorPosição: Recebe por parâmetro um valor inteiro da posição buscada. O método toma a referencia do primeiro nó localizada no atributo início na variável “noDaVez” e com isso inicializa a variável “posicaoDaVez” para 0 para posterior comparação de índices com a posição buscada. Enquanto a referencia do “noDaVez” não for nula, é verificado se o valor inteiro buscado é igual ao atributo inteiro “indice” no nó da lista percorrida. Quando encontrado, o método retorna o nó com o índice procurado, mas caso não, retorna null. Para percorrer a lista, a variável “noDaVez” toma, a cada iteração, a referencia do próximo nó contida no nó atual e incrementa a variável “posicaoDaVez”.

- Remove: Esse método recebe por parâmetro um valor inteiro referente a posição do nó a ser removido da lista. Para isso, utiliza o método “buscarPorPosicao” passando o valor inteiro recebido para acessar o no em questão e “desconectá-lo” de seu anterior e sucessor na lista. Caso não encontrada a posição, o retorno será null, mas caso sim é retornado uma instancia da classe No e verificado se o mesmo é o primeiro, se sim, se é o único elemento; se há apenas 2 e então é necessário atualizar seu sucessor para inicio e fim da lista; e se não corresponder a nenhuma dessas situações, a referencia do atributo inicio é atualizada para seu sucessor apenas. Caso o nó for o ultimo, necessário atualizar o atributo fim da lista para o seu anterior, e se caso não for o primeiro ou ultimo, é tomada a referencia de seu anterior e sucessor e atribuído os atributos de anterior e próximo entre si. Após a remoção, o método invoca o método “atualizaIndice” para atualizar os atributos “indice” dos nós na lista e retorna o nó removido.
- Comprimento: Este método não recebe parâmetro e tem a função de percorrer a lista tendo como referencia a referencia no atributo inicio e incrementando um contador a cada iteração e ao fim retorna o tamanho da lista.
- Imprimir: Este método não recebe parâmetro e também toma como referencia a referencia no atributo inicio para percorrer a lista e imprimir o conteúdo de cada nó.

Classe No

Esta classe é constituída pelos atributos públicos da Classe No “proximo” e “anterior” e têm a referencia do anterior e próximo nó a cada instancia na lista em que for inserido; um atributo do tipo inteiro “índice” que guarda a posição do nó na lista a começar por zero, e um atributo da classe Entidade que tem o intuito de ser genérico e armazenar a referência do conteúdo do nó. Esta classe possui também um construtor que recebe como parâmetro uma instancia da Classe Entidade.

Classe entidade

Esta classe é composta pelo atributo privado de classe “Peca” e possui um construtor que recebe por parâmetro uma instancia da classe Peca (referência à

“peça” devido ao jogo de dominó) e um método getter para acesso e retorno do objeto Peca contido na instancia.

Classe Peca

Esta classe possui 2 atributos do tipo inteiro “n1” e “n2” que armazenam um valor inteiro da peça de dominó. A classe possuem também um construtor que recebe como parâmetro dois valores inteiros “n1” e “n2”, métodos getters e setter para coleta e atribuição de valor aos atributos “n1” e “n2”, e o método “toString” para impressão dos valores da peça.

Classe Dominó

Esta classe pretende simular um jogo de Dominó entre o usuário e o computador. Esta é constituída pelos 4 atributos privados de Classe Lista: “pecas”, armazena inicialmente todas as peças de domino do jogo e depois apenas as peças para compra; “pecasComputador” e “pecasJogador” que armazena as peças após a distribuição das peças para cada jogador, e “pecasJogadas” que armazena todas as peças jogadas. A classe também tem as variáveis globais para fins de controle e são elas:

- “inicioFim” de classe Lista que armazena sempre uma única peça e contem em seu atributo “n1” o atributo “n1” do primeiro elemento da lista “pecasJogadas” e em seu atributo “n2”, o atributo “n2” do último elemento da mesma lista.
- “onOff” do tipo booleando e iniciliazada com true indicando o início do jogo. O valor é alterado para false quando há um vencedor, um empate ou desistência do jogador.
- “vez” também do tipo booleano e tem o propósito e auxiliar na alternância entre os jogadores. O valor true indica a vez do computador e false para o jogador.
- “contadorCompraComputador” e “contadorCompraJogador” do tipo inteiro podem apresentar dois valores: 0 ou 1. Quando em 1, indica que o usuário, indicado no próprio nome da variável, acabou de de realizar a compra de peça e não pode realizar outra compra em seguida. Quando em 0, indica que podem realizar uma compra.

- “vencedor” também do tipo booleano indica se há ou não um vencedor do jogo.
- “contadorEmpateComputador” e “contadorEmpateJogador” do tipo booleano sinalizam que os usuários representados no próprio nome da variável passou sua vez e assim o código interpreta que este não tem peças compatíveis com o jogo. É uma variável que auxilia na identificação de empate.

Os métodos que compõe a classe são:

- Domino: Este método é o construtor da classe e quando instanciado um objeto são geradas instancias da classe Peca que representam as peças do jogo através de duas estruturas de repetição aninhadas e ao final as peças são atribuídas a lista de cada jogador e invocado o método “inicio” para inicialização do jogo.
- Inicio: Este método invoca sequencialmente o método “separaPecas”, que separa aleatoriamente 7 peças para cada jogador; “primeiraPeca”, que identifica qual jogador tem a maior peça dupla ou a maior peça não-dupla para iniciar o jogo; e “mesa” que expõe as peças jogadas e do jogador além de interagir com o usuário.
- SeparaPecas: Este método instancia um objeto da Classe Random, gera um número aleatório entre 0 e o comprimento da lista de peças e percorre a lista “pecas” buscando pelo índice por meio da “buscaPorPosição” da classe lista. Quando encontrado, o nó retornado é adicionado a lista “pecasComputador”. Esse processo é o mesmo para o jogador e ocorre 7 vezes para coletar 7 peças e ao fim inseridas na lista “pecasJogador”.
- PrimeiraPeca: Este método busca pela maior peça dupla ou maior peça não dupla na listas de peças do jogador e do computador, as compara e a maior entre elas é usada para iniciar o jogo.
- Mesa: imprime as peças jogadas e as do jogador. Quanto as peças do computador, é mostrado apenas a quantidade de peças restantes a cada jogada. Este método também é o responsável por delegar a vez de cada jogador e verificar se há um vencedor, empate e se o jogo ainda está ativo.
- Menu: esse método é comumente invocado pelo método “mesa” para interegir com o usuário e apresentar um menu com opções numéricas entre 1, 2, 3 e 9

para, repectivamente, “passar a vez”, “comprar”, “inserir a posição da peça” e “encerrar jogo”.

- InserirPeca: Este método tem a funcionalidade tem remover a peça da lista de peças do jogador e adicionar a lista de “pecasJogadas”. O método é voltado para o jogador e este informa a posição, começando pelo 1, da peça que deseja inserir ao jogo. Após a leitura do console, é feita a busca da peça na lista de peças do jogador e validação da peça por meio do método “validacaoPeca” que retorna se a peça é elegível para o jogo atual ou não. Caso sim, é invocado o meotodo “compatibilidadePeca” passando como paramentro a peça a ser inserida ao jogo e verificado qual das duas últimas peças nos sentidos opostos do jogo a peça a ser inserida é compatível e a adiciona. Após execução, a peça é removida da lista do jogador. Caso a peça não seja válida, é impresso uma mensagem indicando a inelegibilidade.
- ValidacaoPeca: Recebe por parâmetro a peça a ser avaliada e retorna um valor booleano caso a peça em análise seja elegível para o jogo ou não. A validação é feita comparando a igualdade entre os atributos n1 e n2 do objeto Peca e os atributos n1 e n2 do primeiro e único objeto elemento Peca da lista “inicioFim”.
- CompatibilidadePeca: este método analisa qual das extremidades do jogo é compatível com a peça escolhida pelo jogador. Este método tem a função importante de detectar a necessidade de “rotacionar” a peça. Isto é, trocar o valores dos atributos n1 e n2 para a peça ser compatível com alguma extremidade do jogo. Ao final do método a lista “inicioFim” é atualizada com uma peça que contem em seu atributo n1 o valor de “n1” da extremidade esquerda do jogo (inicio da lista “pecasJogadas”) e no atributo “n2” o valor de “n2” da extremidade direita do jogo (fim da lista “pecasJogadas”).
- CompraPeca: Enquanto houver peças (comprimento da lista maior que 0) na lista “pecas” este método busca 1 vez aleatoriamente uma peça simulando a compra de peça no jogo comum. Após o acesso, a peça é inserida na lista do jogador e removida da lista “pecas”.
- Computador: Este método simula a logica do jogo para o computador. Neste é buscada uma peça é a elegível para o jogo atual e inserida ao jogo. Caso não encontrada, o código simula uma compra e verifica novamente suas

peças. Mas caso não nenhuma seja elegível novamente, o computador passa sua vez.

- HaUmVencedor: Este método verifica se há um vencedor comparando se o comprimento da lista de peças de um dos jogadores é igual a zero. Assim, que indica que o vencedor é aquele a utilizar todas as suas peças primeiro.
- HaEmpate: Este método compara se o comprimento da lista de peças a comprar (“pecas”) é igual a 0 com as variáveis globais “controleEmpateJogador” e “controleEmpateComputador”. Estas são variáveis auxiliam na detecção de compra de peças e passagem da tentativa indicando que nenhum dos jogadores tem cartas compatíveis, resultando em empate. Assim é apresentada a mensagem “empate” e a variável global “onOff” setada para false sinalizando o encerramento do jogo.