ADS – IFPB – Campus Monteiro
LISTA SEMANAL – PROGRAMAÇÃO II - PROGRAMAÇÃO ORIENTADA A OBJETOS
Prof. Cleyton Caetano de Souza

Como funciona a pontuação extra:

- Se alguma dupla fizer toda a lista – as 10 questões – dentro do tempo da aula: +0,5 para cada aluno dessa equipe.
- Todos os alunos que fizerem o primeiro programa (questão 7), dentro do tempo da aula: +0,15 para cada aluno.
- Todos os alunos que fizerem o segundo programa (questão 10), dentro do tempo aula: +0,15 para cada aluno.
- **Somente as duplas que permanecerem juntas terão direito a pontuação extra**. Se, após começar, algum aluno da dupla desistir ou for embora, o outro aluno não terá direito a pontuação extra.

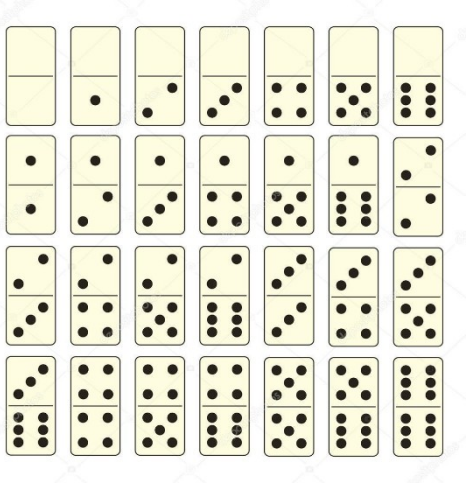
Aula Prática 3 (10/04/23)

1. *Dominó é um jogo de mesa que utiliza peças com formatos retangulares, em que cada um dos lados está marcado por pontos indicando valores numéricos, no intervalo de zero a seis. O termo também é usado para designar individualmente as peças que compõem este jogo.* Crie uma classe chamada “Dominó” (sem acento), com os atributos “lado a” e “lado b”, ambos do tipo inteiro. Use o método set para impedir que esses atributos assumam valores fora do intervalo de zero a seis. Adicione à classe Dominó um construtor para receber os valoreis iniciais dos atributos da peça do Dominó (faça uso do método set, dentro do construtor, para assegurar valores válidos para os atributos). Se um valor inválido for passado (quer seja via construtor, quer seja via método set, o atributo em questão não tem o valor alterado).

2. *No Dominó, para uma peça poder se conectar com outra, elas precisam ter um lado em comum.* Adicione à classe Dominó um método chamado validarJuncao (lê-se validar a junção). Esse método recebe como entrada um objeto do tipo Dominó e retorna um valor booleano, indicando se a peça recebida como parâmetro se conecta ou não com a peça que está chamando o método.

3. Sobrescreva o método toString da classe Dominó para que retorne uma representação da peça seguindo o seguinte padrão: [ladoA, ladoB]. A peça com lado A igual a 4 e lado B igual a 3, por exemplo, deveria retornar no toString [4, 3].

4. Crie uma outra outra classe chamada Jogo, que possuirá um array com capacidade para as 28 peças que compõem o jogo de dominó. Esse array é um atributo, entretanto, ele não possuirá nem um método *getter* e nem um método *setter*. No construtor da classe Jogo, preencha o array com as peças de dominó existentes, conforme a figura ao lado.



5. Adicione à classe Jogo um método chamado embaralhar peças. Esse método não tem parâmetro de entrada e nem retorno. Tudo que ele faz é misturar as peças guardadas dentro do array de dominós. Para isso, você deve sortear dois índices diferentes do array e trocar eles de posição. No método embaralhar peças, faça 100 trocas de peças de lugar.

6. Adicione à classe Jogo um atributo chamado "índice da peça a ser puxada", que começa com o valor 0. Esse atributo também não terá *getter* e nem *setter*. Este atributo vai ajudar na lógica de puxar peças de dominó do jogo. Adicione à classe Jogo um método chamado puxar peça. Esse método não tem parâmetro de entrada e retorna um objeto do tipo peça de dominó. Ele retorna a peça guardada no índice correspondente ao valor atual de "índice da peça a ser puxada" e incrementa o valor desse atributo. Caso todas as peças já tenham sido puxadas, o método retorna null. O método embaralhar deve resetar o valor do atributo "índice da peça a ser puxada" de volta a zero.

Primeiro Jogo – "Unominó"

7. Você vai criar um Jogo de Dominó para apenas uma pessoa. Esse jogo será uma mistura de Uno com Dominó (o "Uno-minó"). No começo da partida, o jogador vai receber algumas peças de dominó aleatórias e uma peça aleatória vai ser sorteada e estará fora da mão do jogador. Essa peça aleatória, fora da mão do jogador, é a "peça da vez" (seria o correspondente a carta do uno que o jogador precisa "casar" com uma da sua mão). Quando o jogador "casa" a peça da vez com uma peça inválida, ele perde o jogo. Quando o jogador consegue "casar" a peça com sucesso, ela sai da sua mão e se torna a nova peça da vez. Na próxima rodada, o usuário tem de "casar" a nova peça da vez com umas das que restou na sua mão (perceba que é diferente do dominó, pois a peça da vez anterior é totalmente esquecida e substituída pela que o jogador escolheu na rodada anterior). O jogador só ganha o jogo se ficar sem peças de dominó na mão. Comece criando um objeto do tipo Jogo de Dominó e chamando o método embaralhar para garantir que as peças estejam em uma ordem aleatória. Use o método puxar peça para dar 6 peças para o usuário. Guarde as peças do usuário em um array. Em seguida, o programa deve sortear outra peça que será a primeira peça do jogo, com a qual o usuário deve conectar as peças que ele tem na mão. Informe ao usuário qual a peça foi sorteada e quais peças se encontram na sua mão. Depois, pergunte ao usuário qual peça da sua mão ele deseja "casar" com a que está na vez. Caso o jogador consiga "casar", a peça de

dominó jogada se torna a peça da vez e ela sai da mão do usuário. Caso o jogador não consiga "casar", ele perde o jogo. Ao final, informe quantas peças o usuário conseguiu tirar da sua mão. Considere que o usuário será uma pessoa boa (diferente do professor) e que não fornecerá valores inválidos durante a execução do programa. **Sugestão:** para escrever o conteúdo do array de Dominó, use Arrays.toString.

Console
Peça da vez: [2, 3] Peças na mão: {[2, 2], [4, 5], [2, 6], [1, 3], [6, 6], [3, 5]} Indique o índice da peça que deseja casar (0-5): 3 [1, 3] se conecta com [2, 3] Peça da vez: [1, 3] Peças na mão: {[2, 2], [4, 5], [2, 6], null, [6, 6], [3, 5]} Indique o índice da peça que deseja casar (0-5): 5 [3, 5] se conecta com [1, 3] Peça da vez: [3, 5] Peças na mão: {[2, 2], [4, 5], [2, 6], null, [6, 6], null} Indique o índice da peça que deseja casar (0-5): 1 [4, 5] se conecta com [3, 5] Peça da vez: [4, 5] Peças na mão: {[2, 2], null, [2, 6], null, [6, 6], null} Indique o índice da peça que deseja casar (0-5): 0 [2, 2] não se conecta com [4, 5] Você perdeu. Fim de Jogo. Você fez 3 pontos.

Segundo Jogo – "Clássico"

8. *No Jogo de Dominó clássico, os jogadores alternam os turnos e, após conectar uma peça a outra, apenas o outro lado da peça alocada fica apto a receber novas peças (aquele que não foi conectado). Durante o jogo, cada jogador escolhe entre as duas pontas ativas (ponta esquerda e direita) para ir conectando a sua próxima peça*. Adicione à classe Jogo dois atributos inteiros chamados "ponta ativa da esquerda" e "ponta ativa da direita" (esses atributos devem ser inicializados com o valor -1 e, ao embaralhar as peças, seus valores devem voltar a ser -1). Esses atributos não terão métodos "*setters*", apenas "*getters*".

9. Adicione à classe Jogo um método chamado "conectar a ponta esquerda" que receberá um objeto do tipo Dominó e retornará um valor booleano (indicando se a jogada foi bem sucedida ou não). Se alguma das pontas estiver com valor -1, isso indica que a peça recebida foi a primeira jogada e os seus lados (A e B) devem ser atribuídos a ponta esquerda e direita (respectivamente) e o método retorna **true**. Se esse não for o caso, deve-se testar se algum dos lados da peça recebida se conecta com a ponta esquerda. Se a conexão for possível, o outro lado (aquele que não foi testado) deve ser atribuído a ponta esquerda e o método deve retorna **true**. Se a conexão não for possível, o método retorna **false**. Adicione à classe Jogo um outro método chamado "conectar a ponta direita", que funcionará da mesma forma, mas com a verificação sendo em relação ao atributo "ponta ativa da direita". Por último, adicione à classe Jogo um método estático chamado "teste se acabou", que recebe um array de Dominó e retorna um valor lógico. Se o array estiver apenas com valores **null**, o método retorna **true** (e false, caso contrário).

10. Escreva um programa que simulará um Jogo entre dois jogadores humanos (os usuários). Cada jogador começa com 6 peças, que serão atribuídas aleatoriamente, usando a classe Jogo. Em seguida, puxe uma peça do Dominó para dar início ao jogo. O jogador então deve usar uma das peças da sua mão para se conectar a peça que está em Jogo (após a primeira peça, você deve exibir as peças que estão na mão do Jogador, você deve perguntar se ele quer conectar com a ponta esquerda ou direita do Jogo atual). Se um dos jogadores falhar em tentar conectar uma peça ou algum dos jogadores jogar todas as peças que dispõem (use o método teste se acabou), o jogo termina. Quando o jogo acabar, informe qual jogador venceu.

Console

Peça sorteada para iniciar o jogo: [1, 1]

Em jogo: esquerda: 1 | direita: 1
Mão do Jogador 1: {[2, 2], [4, 5], [2, 6], [1, 3], [6, 6], [3, 5]}
Jogador 1, indique o índice da peça que deseja jogar: 3
Jogador 1, em qual lado deseja conectar? direito

Em jogo: esquerda: 1 | direita: 3
Mão do Jogador 2: {[0, 2], [3, 4], [4, 4], [1, 5], [2, 6], [5, 5]}
Jogador 2, indique o índice da peça que deseja jogar: 1
Jogador 2, em qual lado deseja conectar? direito

Em jogo: esquerda: 1 | direita: 4
Mão do Jogador 1: {[2, 2], [4, 5], [2, 6], **null**, [6, 6], [3, 5]}
Jogador 1, indique o índice da peça que deseja jogar: 1
Jogador 1, em qual lado deseja conectar? direito

Em jogo: esquerda: 1 | direita: 5
Mão do Jogador 2: {[0, 2], **null**, [4, 4], [1, 5], [2, 6], [5, 5]}
Jogador 2, indique o índice da peça que deseja jogar: 3
Jogador 2, em qual lado deseja conectar? esquerdo

Em jogo: esquerda: 5 | direita: 5
Mão do Jogador 1: {[2, 2], **null**, [2, 6], **null**, [6, 6], [3, 5]}
Jogador 1, indique o índice da peça que deseja jogar: 5
Jogador 1, em qual lado deseja conectar? direito

Em jogo: esquerda: 3 | direita: 5
Mão do Jogador 2: {[0, 2], **null**, [4, 4], **null**, [2, 6], [5, 5]}
Jogador 2, indique o índice da peça que deseja jogar: 5
Jogador 2, em qual lado deseja conectar? direito

Em jogo: esquerda: 3 | direita: 5
Mão do Jogador 1: {[2, 2], **null**, [2, 6], **null**, [6, 6], **null**}
Jogador 1, indique o índice da peça que deseja jogar: 1
Jogador 1, em qual lado deseja conectar? direito
Não conectou. Jogador 1, perdeu.

Fim de Jogo.