PRONTUÁRIO: INDICAR NO ZIP Dia: 14/09/2023 Horário: 19h00 – 22h30

✓ A prova é individual e sem consulta. ✓ Não é permitido utilizar qualquer código implementado por você anteriormente. ✓ Não coloque seu nome no projeto. ✓ Atribui-se nota zero à prova em desacordo com os itens acima.	✓ A prova deve ser nomeada da seguinte forma: PRONTUARIO_P1, com o "SC". ✓ Não envie apenas as classes, mas todo o projeto. ✓ Envie o projeto como zip no Moodle.

Considere o contexto a seguir:

Após estudar Programação Orientada a Objetos você percebeu que é capaz de modelar qualquer coisa, até um jogo de Truco. Para se divertirem, você e seu colega começaram a implementar a base do jogo. Seu colega já fez as regras de negócio que representam cartas de um baralho e também um deck de 40 cartas. Você vai implementar classes representando os jogadores (Player), o jogo (Game), as mãos do jogo (Hand) e as rodadas de uma mão (Round). Para isso, você deve utilizar como base o diagrama UML descrito a seguir:

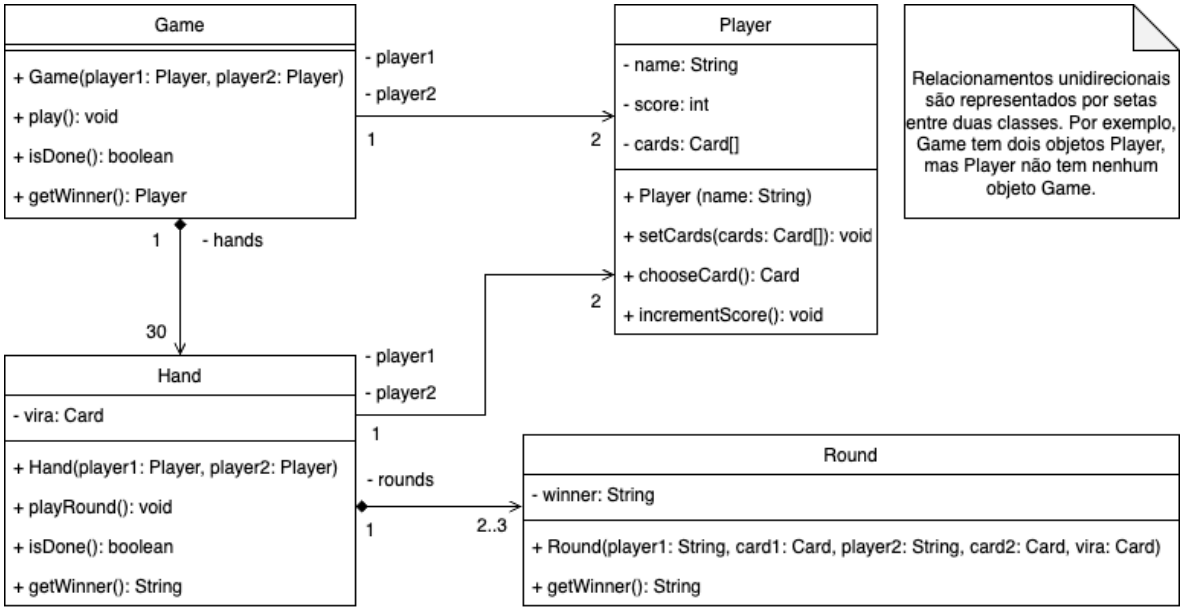

Figura 1 - Modelo do jogo de truco

O jogo acontece entre dois jogadores, sendo o vencedor aquele que fizer primeiro doze pontos. Nesta primeira versão, não haverá aumento de pontos e o vencedor de cada mão ganhará um ponto. Para saber quem é vencedor de uma mão, os jogadores disputam quem é o melhor de três rodadas. No começo de uma mão, cada jogador recebe três cartas e uma carta é aberta na mesa, chamada "vira". A cada rodada, cada jogador joga uma carta, sendo o vencedor aquele que jogar a carta mais forte, tendo como base a vira. Você não precisa se preocupar em implementar a lógica da carta mais forte, pois seu amigo já implementou. Graças ao princípio do encapsulamento, tudo o que você precisa saber é que ao invocar o método *compareValueTo* de um objeto carta A, passando outra carta B e a vira como parâmetros, o método retornará um número positivo se a carta A for mais forte, um número negativo se a carta A mais fraca, e zero se as cartas A e B tiverem o mesmo valor. Como há a possibilidade de empate em uma ou mais rodadas, existem as seguintes regras para determinar um vencedor da melhor de três em uma mão:

- Se algum jogador ganhar duas rodadas, ele ganha a mão;
- Se empatar na primeira rodada, quem ganhar a segunda vence a mão;
- Se empatar na segunda rodada, quem ganhou a primeira vence a mão;
- Se empatar na primeira e segunda rodada, quem fizer a terceira vence a mão;
- Se empatar na terceira rodada, quem ganhou a primeira vence a mão;
- Se todas as três rodadas empatarem, ninguém ganha ponto.

Execute as atividades a seguir para a implementação do exercício em Java. Para a atribuição da nota será levada em conta não apenas a funcionalidade, mas a qualidade, adequação e pertinência de cada solução. Bom senso faz parte da prova.

#	Descrição	Pont.
1	Crie as classes do modelo utilizando os modificadores de acesso mais adequados aos métodos e atributos, segundo o conceito de encapsulamento.	0,5pt
2	Estruture as associações de composição das classes do modelo. Utilize arrays para armazenar os relacionamentos envolvendo múltiplos objetos.	0,5pt
3	Implemente os métodos da classe Player previstos no diagrama. O método *setCards(Card[] cards)* serve para receber um array com as cartas do jogador no início de cada mão. O método *chooseCard()* retorna uma das cartas, não deixando-a disponível para ser jogada novamente durante a mesma mão.	1,5pt
4	Implemente os métodos da classe Round previstos no diagrama. O vencedor da mão deve ser definido dentro do próprio método construtor, para que fique disponível para consulta por meio do método *getWinner()*. O método *getWinner()* deve retornar o nome do vencedor ou *null*, caso haja empate.	1,0pt
5	Implemente o construtor da classe Hand, que ao ser invocado, cria um Deck e o embaralha (método *shuffle()*), pega uma carta do deck como vira e dá três cartas a cada jogador. Para obter cartas do Deck, use os métodos *takeOne()* ou *take(int numberOfCards)*, que obtém uma ou múltiplas cartas, respetivamente.	1,0pt
6	Crie o método *playRound()* da classe Hand, que cria um objeto Round a partir de cartas obtidas pelo método *chooseCard()* de cada Player. Após criar o objeto round, imprima o resultado e guarde o objeto no array de rodadas da mão.	0,5pt
7	Implemente o método *isDone()* da classe Hand, que determina se a mão já foi encerrada a partir dos resultados das rodadas já jogadas nessa mão.	1,0pt
8	Implemente o método *getWinner()* da classe Hand, que retorna o nome do vencedor da rodada ou *null* se a rodada não terminou ou terminou empatada.	1,0pt
9	Crie o construtor da classe Game, que inicializa o jogo com a primeira rodada e a deixa pronta para que seja jogada.	0,5pt
10	Crie o método *play()* da classe Game, que invoca o método *playRound*() da mão em curso. Se a mão já estiver concluída, o método primeiro contabiliza o ponto da mão encerrada, apresenta o resultado no console, cria uma nova mão e só então invoca o método *playRound()* dessa mão.	1,0pt
11	Implemente o método *isDone()* da classe Game, que determina se o jogo já foi concluído.	0,5pt

12	Implemente o método *getWinner()* da classe Game, que retorna a instância do jogador vencedor ou *null* se o jogo não estiver encerrado.	0,5pt
13	Crie uma classe chamada Principal, que contenha um método *main().* Dentro deste método, crie dois jogadores, um jogo e invoque o método *play()* até o jogo se encerrar. Ao final do jogo, imprima o nome do vencedor.	0,5pt

Prêmio Usain Bolt: O aluno que terminar todas as atividades corretamente primeiro ganha 1pt adicional para usar em outra prova. Você está voando?

***** Boa sorte! *****