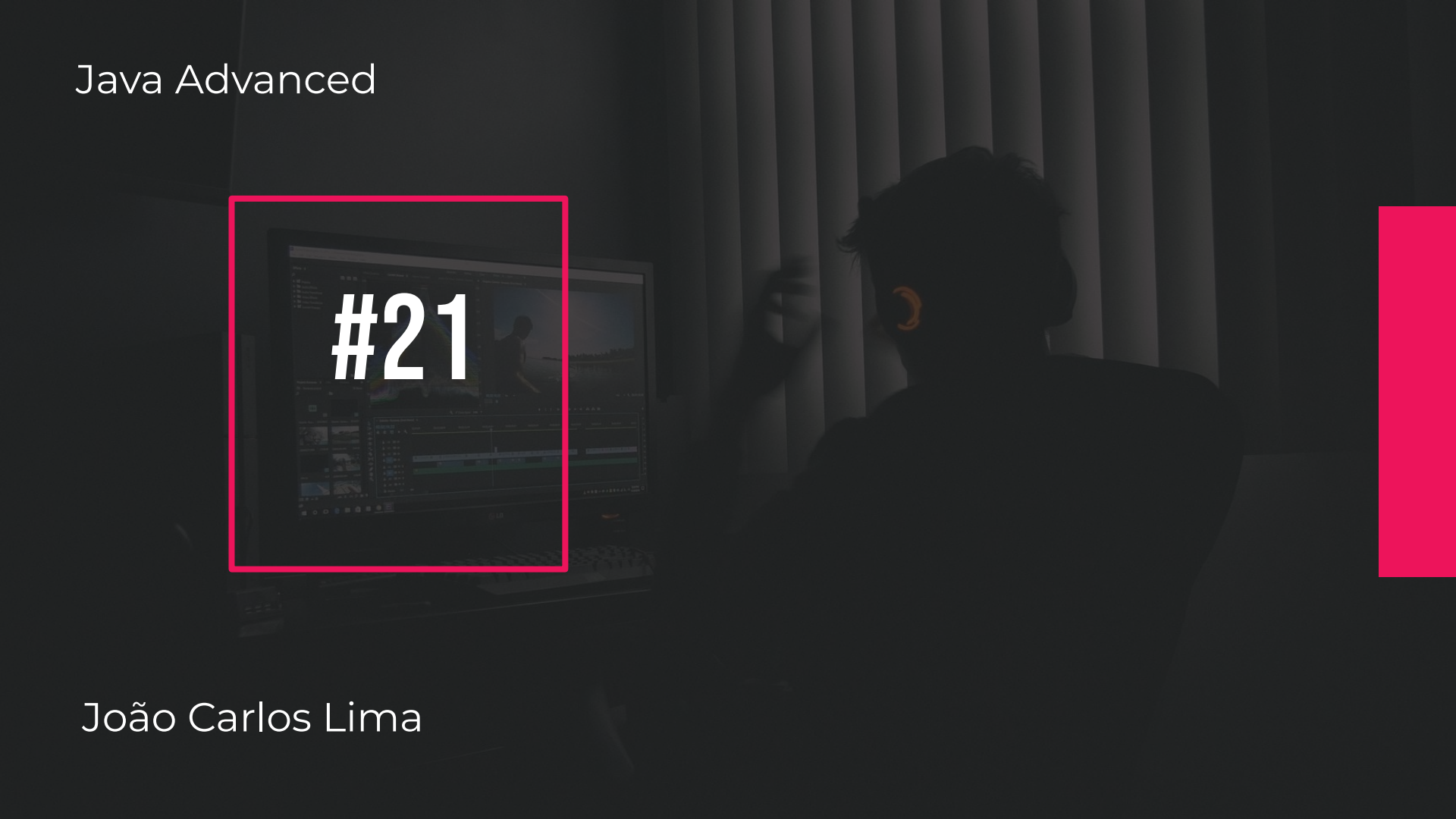


Java Advanced



#21

João Carlos Lima

INTRODUÇÃO



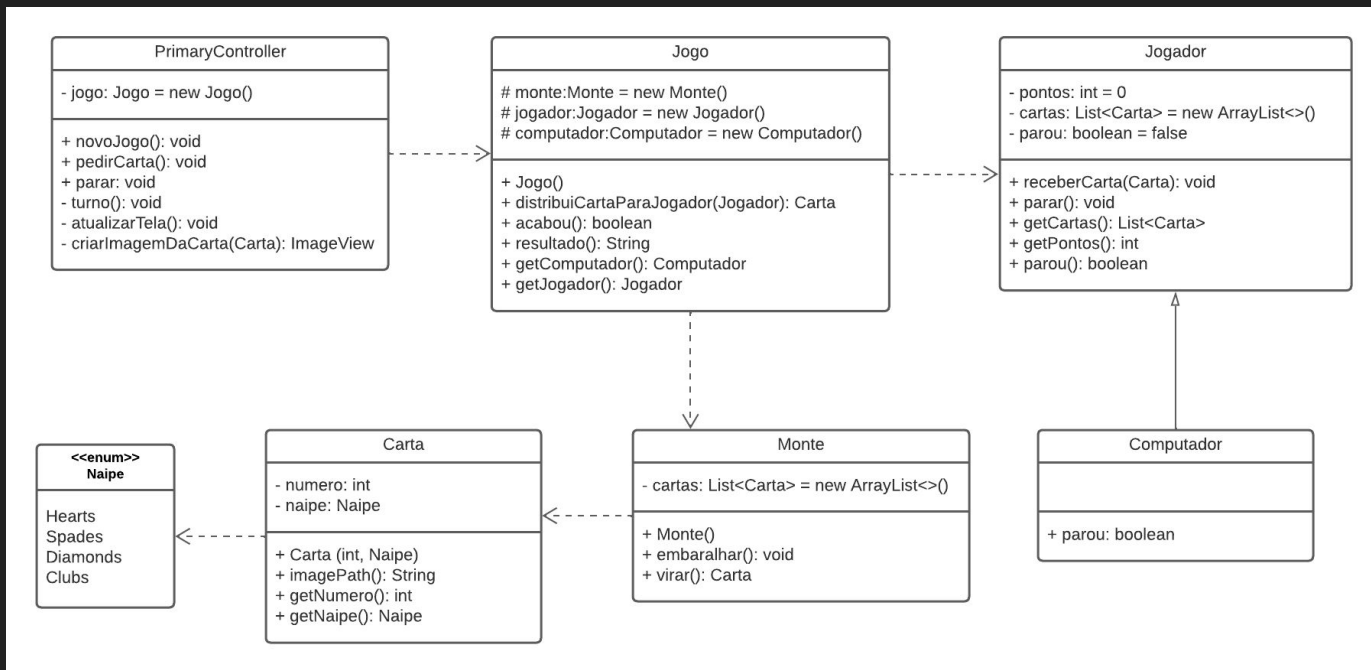
O objetivo deste projeto é criar um jogo de **21** completo com interface gráfica

INTRODUÇÃO



DIAGRAMA DE CLASSES

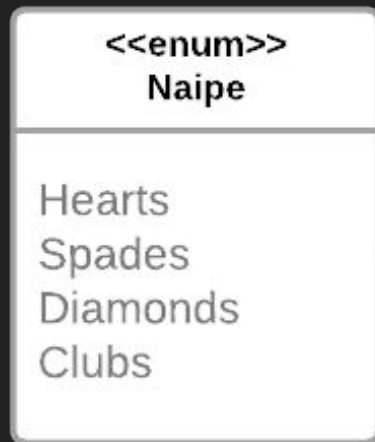
O diagrama representa as classes e os relacionamentos entre elas.
Crie o código java que representa o diagrama a seguir.



ENUM NAIPE

Essa classe é um ENUM com os naipes do jogo

O desenvolvimento ficará mais fácil se utilizarmos os nomes em inglês como está definido no nome dos arquivos das imagens.



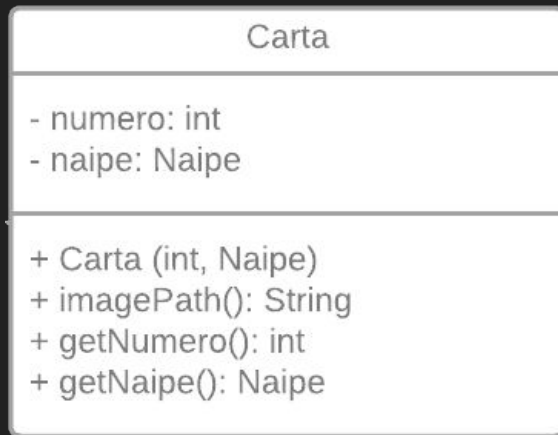
CLASSE CARTA

Essa classe representa uma carta do jogo.

- O método `imagePath` deve retornar o caminho da imagem que representa a carta. Observe os nomes dos arquivos das imagens das cartas para implementar esse método. Ex:



```
return "classic-cards/" + numero + this.naipe + ".png";
```



CLASSE JOGADOR

Essa classe representa o jogador do jogo. O jogador pode ser o usuário ou o computador.

- o método `receberCarta` deve adicionar uma carta à mão do jogador e acrescentar a pontuação desta carta ao jogador
- os demais métodos são auto explicativos

Jogador
<ul style="list-style-type: none">- pontos: int = 0- cartas: List<Carta> = new ArrayList<>()- parou: boolean = false
<ul style="list-style-type: none">+ receberCarta(Carta): void+ parar(): void+ getCartas(): List<Carta>+ getPontos(): int+ parou(): boolean

CLASSE COMPUTADOR

Essa classe representa o adversário do usuário.

- A classe computador é um tipo específico de jogador
- Essa classe sobreescreve o método parou
- O computador sempre para quando tem mais de 16 pontos.



CLASSE MONTE

Essa classe representa o monte de cartas da partida.

- Ao ser criado, o monte deve criar todas as cartas do baralho e adicionar a lista de cartas.
- O método embaralhar deve deixar a lista de cartas em ordem aleatória.
- O método virar deve remover a primeira carta do monte e retornar a carta virada.



CLASSE JOGO

- Quando o objeto jogo for criada, o monte deve ser embaralhado.
- O método que distribui uma carta para um jogador, deve virar uma carta do monte e entregar para o jogador recebido como parâmetro. Se o jogador já tiver parado, não entregue carta para o jogador e retorne null.
- O método acabou deve verificar se o jogo terminou, considerando se os dois jogadores pararam ou se algum jogador passou de 21 pontos.
- O método resultado deve retornar um texto com o resultado da partida (“Você perdeu”, “Você ganhou” ou “Empate”)

Jogo

```
# monte:Monte = new Monte()  
# jogador:Jogador = new Jogador()  
# computador:Computador = new Computador()
```

```
+ Jogo()  
+ distribuiCartaParaJogador(Jogador): Carta  
+ acabou(): boolean  
+ resultado(): String  
+ getComputador(): Computador  
+ getJogador(): Jogador
```

CONTROLLER

Essa classe faz a ligação da View com os models. Os métodos públicos são acionados pela interface gráfica.

- O método novoJogo deve limpar todos os elementos de dados e visuais e deixar o jogo pronto para uma nova rodada.
- O método pedir carta deve iniciar um turno
- O método parar deve chamar o método parar do jogador e iniciar um turno
- O método turno deve distribuir uma carta para os jogadores que não pararam e verificar se o jogo acabou
- O método atualizar deve atualizar os elementos da tela (labels e imagens)

PrimaryController
- jogo: Jogo = new Jogo()
+ novoJogo(): void + pedirCarta(): void + parar: void - turno(): void - atualizarTela(): void - criarImagemDaCarta(Carta): ImageView