

ECM251 - Linguagens de Programação I

11 - Revisão de P00

Prof. Murilo Zanini de Carvalho

Objetivos da aula de hoje

- Deixar clara a diferença entre a abordagem procedural e a abordagem orientada a objetos.
- Estudar os conceitos fundamentais do paradigma orientado a objetos.
- Reescrever programas com orientação procedural para orientação a objetos.
- Retomar alguns conceitos de Java.

Um *food truck* e um restaurante, analogia de paradigmas



Um *food truck* e um restaurante, analogia de paradigmas

Em um food truck, todas as tarefas ficam por conta do funcionário que está trabalhando ali.

Ele é RESPONSÁVEL por fazer a comida, servir a comida, cobrar e pegar os pedidos com os clientes.

Essa abordagem funciona mas apresenta diversos problemas. Com o aumento do negócio, um número maior de clientes vai se acumulando e fica difícil atender a todos as RESPONSABILIDADES do negócio.

Um *food truck* e um restaurante, analogia de paradigmas

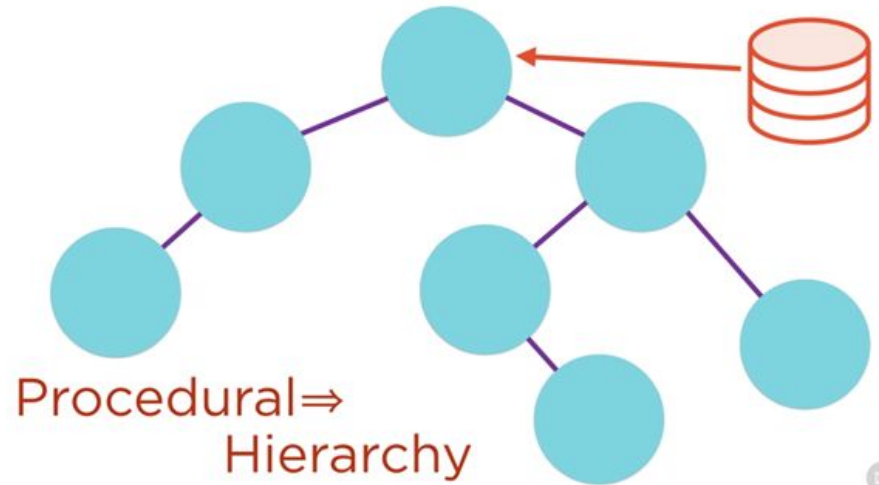
Já em um restaurante, existem diversos TRABALHOS. Cada funcionário fica com um conjunto de RESPONSABILIDADES.

Um garçom fica responsável por pegar os pedidos dos clientes e levar os pratos preparados para a mesa. Os funcionários da cozinha ficam responsáveis por preparar os pedidos que chegam e retornar os pratos com a comida. O caixa faz as cobranças.

Cada funcionário do restaurante realiza uma tarefa que, da sua INTERAÇÃO, faz com que o restaurante funcione.

Na abordagem procedural

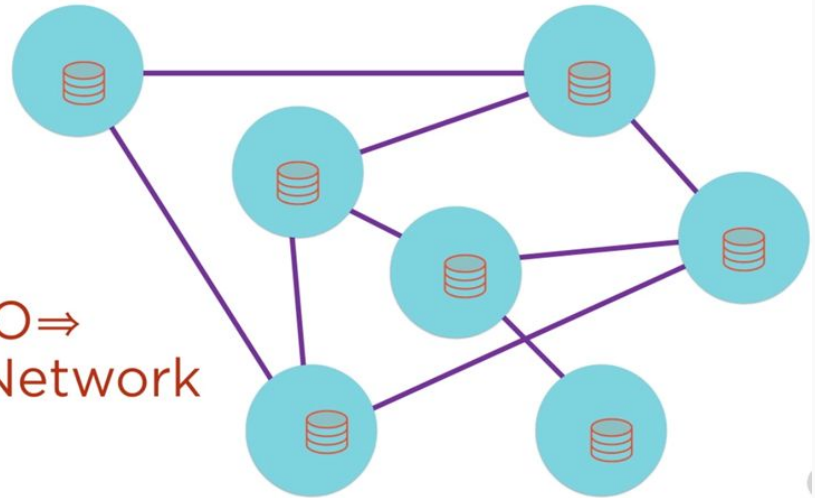
Data centric, e o fluxo de dados é muito importante. O sistema possui acesso a um conjunto de dados e todos os módulos alteram esses dados de alguma forma (Sanches, 2018).



Na abordagem orientada a objetos

OO não é data centric, parece um rede.
Cada objeto será responsável por gerenciar o seu próprio conjunto de dados (Sanches, 2018).

OO \Rightarrow
Network



Pilares da Orientação a Objetos

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

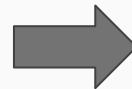
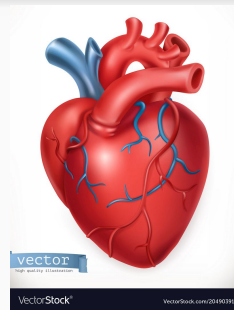


Retirado de
(<https://www.vivaxsolutions.com/images/four-pillars.png>), em
01/08/2019

Abstração

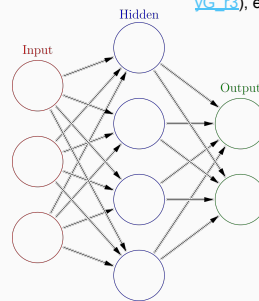
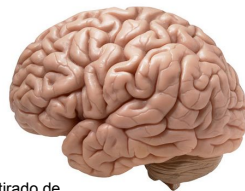
Os pontos fundamentais de um objeto ou problema são listados para representar esse objeto ou problema.

Quando essa representação fica muito complexa, ela deve ser dividida em abstrações mais simples. Cada abstração deve ter uma funcionalidade limitada, dessa forma, é possível gerenciar suas ações e seu impacto no projeto.



Retirado de
(<https://cdn2.vectorstock.com/11000x1000/03/91/human-heart-medical-internal-organs-3d-icon-vector-20490391.jpg>), em 01/08/2019

Retirado de
(https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRSXKF4VaQV5UUXP6JDiiQeJB-arF9dhVxsKCNa1R3Hsi_vY_r3), em 01/08/2019



Retirado de
(<https://cdn.the-scientist.com/assets/article/No/36663/llmq/15248/d305ec2a-9f5a-4894-8cd3-a7c43bb0756b-brain-640.jpg>), em 01/08/2019

Retirado de
(https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/1200px-Colored_neural_network.svg.png), em 01/08/2019

Encapsulamento

Quando um garçom vai fazer um pedido para o chef, ele não deve dizer ao chefe como ele deve cozinhar, ele apenas deve informar o que foi pedido. Cada classe é responsável por suas tarefas. Ainda assim, é necessário que o garçom possa fazer o pedido para o chef.

Para essa função, existe o local onde ele deve fazer esse pedido e receber de volta do chef o prato quando ele estiver pronto. Esse conceito é o da interface pública do objeto.



Retirado de
(<https://c8.alamy.com/comp/DX8DP7/chef-at-the-pass-plating-food-michelin-starred-the-three-chimneys-DX8DP7.jpg>), em 01/08/2019

Herança

Quando as abstrações estão sendo construídas, podem existir métodos comuns em diversas classes propostas. Para retirar essa duplicação de código e para concentrar essas características comuns, é possível utilizar a herança.

Quando um objeto herda do outro, ele recebe seus métodos e atributos, tornando se também um objeto daquela classe.



Polimorfismo

Quando herdamos as características de um classe pai, alguns comportamentos podem precisar ser sobrescritos. Isso traz uma personalização para a classe filho.

Quando o método personalizado for chamado de uma instância da classe filha, ele vai ser chamado. Quando a chamada vier de uma instância da classe pai, o método original vai ser chamado.



Retirado de
(<https://i.pinimg.com/originals/d4/6b/bc/d46bbc09f471eb370ccdf30aae8ae9a9.png>), em 01/08/2019

Programas Orientados a Objeto

- Os objetos vão se conversar através de mensagens. Essas mensagens podem ser implementadas através de métodos, mas não é obrigatório.
- Métodos são funções que manuseiam as mensagens entrantes.
- Alguns métodos podem utilizar funções para fazer trabalhos simples de decodificação, cálculo ou algo do tipo.

Messages
**objects send
messages to
one another**

Methods
handle messages

Functions
do arbitrary work

Objetos

- Objetos são definidos pelo o que eles fazem, não pelo o que eles contêm. Eles devem ser vistos como uma caixa preta, que você pede para eles fazem algumas operações, e você não sabe qual a implementação e qual o processamento que ele irá realizar. Então podemos dizer que os objetos possuem responsabilidades, e devem realizar operações com coesão.
- Então não sabemos como o objeto funciona, mas sabemos como pedir coisas para eles, bem como o que esperar como resposta. O que existe dentro do objeto deve ser desconhecido para nós, como uma caixa preta.

Objects are defined by
what they do, not what
they contain.

They have responsibilities.

**Data Abstraction,
Implementation Hiding**



Objetos - Princípio de Responsabilidade Única

Cada objeto deve possuir apenas uma responsabilidade. As duas imagens abaixo violam esse princípio.



Single Responsibility



Objetos - Acoplamento

Se você está pedindo informações para uma classe, isso pode ser indício que o acoplamento está sendo maior do que deveria.

Se precisamos realizar algum processamento com informações que o objeto possui, devemos então pedir para o objeto realizar o processamento e não pegar a informação para tal.



Ask for help,
not for information.

Don't `get()` the data.
Ask the object that
has the data to do
the work for you.

Delegation

Objetos - Isolamento

Devo ser capaz de modificar qualquer variável, tipo ou implementação da minha classe, sem que as que utilizam ela se quer notem a diferença.

You should be able to radically change the implementation of a class without impacting the clients.

Projeto de Hoje

Vamos implementar um jogo de P@ker.



Retirado de
(<https://blog.royalpag.com/wp-content/uploads/2017/05/86711-voce-sabe-como-estudar-poker-corr-etamente-999x640.jpg>), em 11/08/2019



Retirado de
(<https://www.planocritico.com/wp-content/uploads/2012/10/casino-royale-james-bond-007-plano-critico.jpg>), 11/08/2019

Etapa 1

Modele as classes Card e Deck.

Atenção para as definições feitas até aqui.

O testes dessas classes, apenas nessa etapa, pode ser feito na classe Main.

Etapa 2

Refatore seu código para adicionar uma classe Dealer.

O Dealer deve ser responsável por distribuir as mãos dos jogadores.

Cada mão do jogador deve possuir 5 cartas. Implemente a mão utilizando a classe Hand (que você deve modelar).

Todo o jogo deve acontecer quando o método play() da classe Game for chamado.

Etapa 3

Refatore a classe Hand, ela deve conseguir verificar se as cartas que ela tem na mão formam uma das jogadas:

- Pair
- Three of a Kind





































POKER HANDS (Ace can be either lowest or highest ranking card)

A♥	K♥	Q♥	J♥	10♥	Royal Flush Highest-ranking straight flush
5♥	4♥	3♥	2♥	A♥	Straight Flush 5 cards in rank order, same suit
4	4	4	4		Four of a Kind 4 same rank
3	3	3	2	2	Full House 3 same rank + 2 same rank
♥	♥	♥	♥	♥	Flush 5 cards same suit
5	4	3	2	A	Straight 5 cards in rank order
3	3	3			Three of a Kind 3 same rank
A	A	2	2		Two Pair 2 same rank, twice
2	2				Pair 2 same rank
A					High Card Highest ranking card, no hands

Etapa 4

Refatore a classe Hand, para implementar o restante das mãos possíveis.

POKER HANDS (Ace can be either lowest or highest ranking card)

					Royal Flush Highest-ranking straight flush
					Straight Flush 5 cards in rank order, same suit
					Four of a Kind 4 same rank
					Full House 3 same rank + 2 same rank
					Flush 5 cards same suit
					Straight 5 cards in rank order
					Three of a Kind 3 same rank
					Two Pair 2 same rank, twice
					Pair 2 same rank
					High Card Highest ranking card, no hands

Etapa 5

- Implemente na classe Game, duas mãos e verifique quem foi o jogador vitorioso.








































POKER HANDS (Ace can be either lowest or highest ranking card)

A	K	Q	J	10	Royal Flush Highest-ranking straight flush
5	4	3	2	A	Straight Flush 5 cards in rank order, same suit
4	4	4	4		Four of a Kind 4 same rank
3	3	3	2	2	Full House 3 same rank + 2 same rank
♥	♥	♥	♥	♥	Flush 5 cards same suit
5	4	3	2	A	Straight 5 cards in rank order
3	3	3			Three of a Kind 3 same rank
A	A	2	2		Two Pair 2 same rank, twice
2	2				Pair 2 same rank
A					High Card Highest ranking card, no hands

Etapa 6

- Torne a quantidade de jogadores dinâmica (máximo de 9 jogadores), permitindo que o usuário determine quantidade de jogadores.

POKER HANDS (Ace can be either lowest or highest ranking card)






































					Royal Flush Highest-ranking straight flush
					Straight Flush 5 cards in rank order, same suit
					Four of a Kind 4 same rank
					Full House 3 same rank + 2 same rank
					Flush 5 cards same suit
					Straight 5 cards in rank order
					Three of a Kind 3 same rank
					Two Pair 2 same rank, twice
					Pair 2 same rank
					High Card Highest ranking card, no hands

Strength ↑

Etapa 7

- Crie uma classe para representar cada jogador (Player) e permita que o usuário informe o nome de cada jogador.

POKER HANDS (Ace can be either lowest or highest ranking card)






































					Royal Flush Highest-ranking straight flush
					Straight Flush 5 cards in rank order, same suit
					Four of a Kind 4 same rank
					Full House 3 same rank + 2 same rank
					Flush 5 cards same suit
					Straight 5 cards in rank order
					Three of a Kind 3 same rank
					Two Pair 2 same rank, twice
					Pair 2 same rank
					High Card Highest ranking card, no hands

Strength ↑

Desafio 1

- Incremente o jogo para que a mesa possa dar as duas cartas e verificar quem foi o jogador vitorioso.

POKER HANDS (Ace can be either lowest or highest ranking card)

					Royal Flush Highest-ranking straight flush
					Straight Flush 5 cards in rank order, same suit
					Four of a Kind 4 same rank
					Full House 3 same rank + 2 same rank
					Flush 5 cards same suit
					Straight 5 cards in rank order
					Three of a Kind 3 same rank
					Two Pair 2 same rank, twice
					Pair 2 same rank
					High Card Highest ranking card, no hands

Strength ↑