POO | Programação Orientada a Objetos

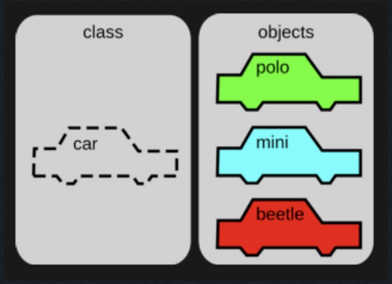
O que é um paradigma?

Um conjunto de formas que servem de modelo;
Um padrão;

Quais são os principais problemas que a POO resolve?

Entre muitos outros problemas, os principais que podemos destacar, seriam:

- Desorganização e dificuldade de manutenção;
- Repetição;
- Dificuldade de reutilização de código;



Classe

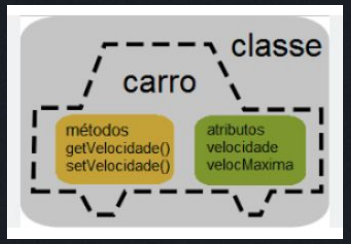
É um modelo/molde de algo do mundo real em que necessitamos virtualizar (trazer para nosso código);

Ela possui características e ações;

Objeto

Quando realizamos a instância de nossa classe, criamos um objeto;

Obrigatoriamente, vai ter as características e métodos que o seu "molde" pré-determina;



Atributos

Características que minha classe predetermina e que meu objeto terá, obrigatoriamente;

(Somente o que for necessário ao meu código);

Métodos

Ações que poderão ser executadas;

Podem ou não receber argumentos, assim como os seus retornos podem ou não existir;

O padrão adotado para sua nomeação, geralmente, são nomes de ações executadas;

De forma resumida e teoricamente, quais são 4 pilares da POO?

Abstração

(Somente o necessário)

Encapsulamento

(Proteção dos dados internos)

Herança

(Reuso de código em implementações filhas)

Polimorfismo

(Mesmo método, implementações diferentes)

Uso da palavra reservada NEW

Sempre que formos fazer uma instância de uma classe, utilizaremos a palavra reservada NEW para criar a mesma;

JavaScript

```
class Animal { }

const animal = new Animal();
```

Uso da palavra reservada THIS

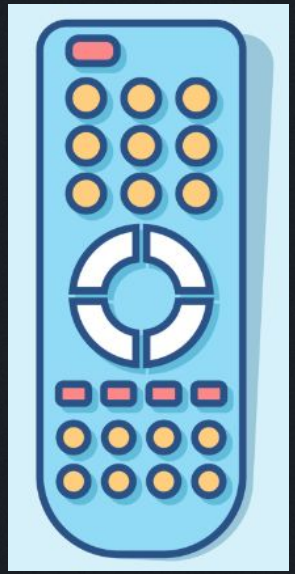
Quando quisermos acessar, INTERNAMENTE, os dados da minha classe, eu preciso fazer o uso da palavra reservada, seguida da informação desejada;

```
class Animal {
    nome;
    idade;
    som;

    fazerAniversario() {
        this.idade++;
    }
}
```

Abstração

Processo de simplificar complexidades, isolando detalhes irrelevantes e destacando apenas os aspectos essenciais;



Método Construtor

Método especial de uma classe, chamado no momento de sua instância;

Geralmente, utilizado para inicializar valores dos atributos e validá-los;

Mão na massa:

Vocês deverão implementar uma calculadora simples, contendo as 4 operações básicas;

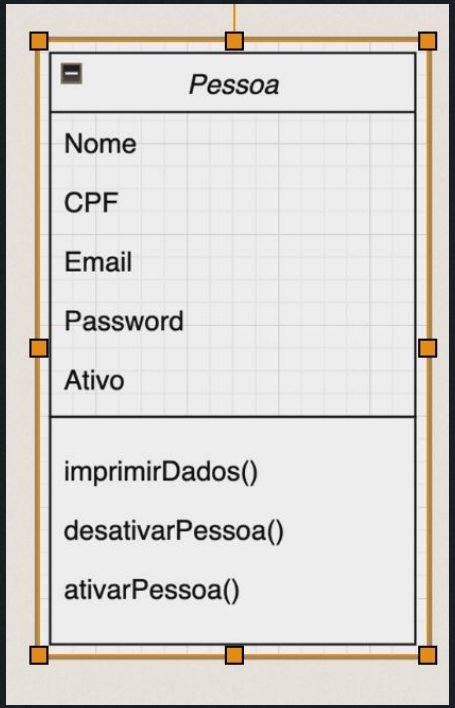
REQUISITOS:

- Quando a calculadora for instanciada, deverá receber 3 argumentos (um número qualquer, uma operação a ser realizada e um segundo número);
- Vocês precisam validar se os argumentos recebidos são válidos;
- A partir disso, demonstram o uso de sua calculadora através do terminal, executando seu script;

UML (Ou, Diagrama de Classes)

Utilizada para modelagem visual;

Diagramas que representam as entidades do sistema, mostrando suas informações contidas e suas devidas ações, relacionamentos com outras classes e, também, a visibilidade de suas informações;



Encapsulamento

Proteção do que é necessário, em nosso código;

Usuário externo só poderá utilizar e/ou modificar o que eu permitir;

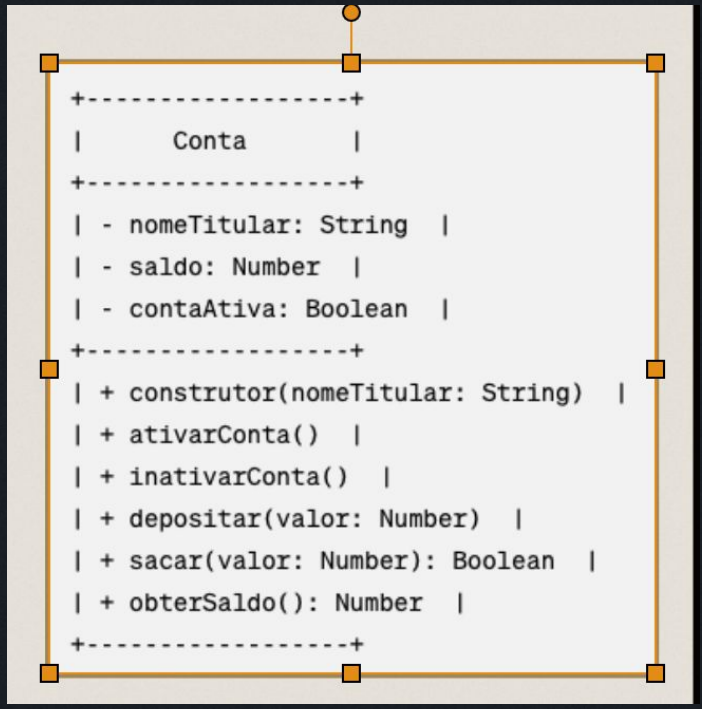
Métodos acessores

Quando estamos trabalhando com encapsulamento, costumamos ter os métodos acessores **getters** e **setters** das nossas propriedades, para que consigamos realizar o acesso e alterarmos os valores do que for permitido e necessário em nossa classe;

POO **Mão na massa:**

REQUISITOS:

- O saldo deve ser inicializado com zero e o atributo contaAtiva como true;
- A classe só poderá ser instanciada se o nome tiver o mínimo de 4 caracteres (não poderá receber números);
- O depositar e o sacar só poderão operar se tiver saldo disponível;
- A ação de inativar uma conta só poderá ser realizada se a conta estiver zerada e, se a mesma tiver ativa;
- Caso a operação (ativar/desativar) tenha sido realizada com exito, deverá retornar um valor booleano indicando isso;
- Faça uso de sua classe em um script;
- Em seu script, crie um array que receba todas as instâncias de conta. Antes de realizar a criação de uma nova conta, verifique se já existe um titular com mesmo nome dentro do mesmo e, só crie a conta caso não exista (dê feedback ao seu usuários sobre a criação ou não);



ada