

Simula 67

Primeira Linguagem Orientada a Objetos

A primeira linguagem orientada a objetos foi desenvolvida nos anos 60 pelos noruegueses ao lado, Ole-Johan Dahl (1931-2002) and Kristen Nygaard (1926-2002).

O Simula 67 continha muitos dos conceitos das principais linguagens OO de hoje, como;

- Classes e objetos;
- Subclasses;
- Métodos virtuais;
- Frameworks;
- Concorrência;
- Gerenciamento de memória.



Fonte: <https://fsharp.org/computing/ModernComputing/Software/Simula.html>

O que é função utilitária e conceitos básicos

Aprendizado tradicional da Orientação a Objetos

Chamamos de objetos a representação programática de algo, como, por exemplo, um carro, onde temos atributos e ações tomadas que podem ser tomadas ou executadas por ele.

Representação da classe carro



Um carro, tem: altura, largura, cor, combustível, cilindradas, número de passageiros e velocidade máxima. Ele pode: ligar, desligar, parar, andar e acelerar.

| Carro |
|---|
| + altura: Numérico + largura: Numérico + cor: Cores + combustível: Combustíveis + cilindradas: Inteiro + passageiros: Inteiro + velocidadeMaxima: Numérico |
| + ligar(): void + desligar(): void + parar(): void + andar(): void + acelerar(): void + velocidadeAtual(): Numérico + estaParado(): Booleano + estaAndando(): Booleano |

Primeira parte **atributos**.
Segunda parte **métodos**.

E a aplicação disso no dia a dia?

O problema da utilização desse tipos de exemplo para o aprendizado é que ele demonstra por analogia como fazer a representação de um objeto concreto em uma classe, mas geralmente o programador que está iniciando com a POO tem dificuldade para transpor-la para um cenário que possa ser realmente aplicado.

- Os padrões atuais da POO são muito mais do que simplesmente transpor objetos do mundo real para o seu código. Esses padrões novo demonstram como componentizar suas aplicações, de forma a aumentar o recuso de código e facilitar a manutenção futura.

Classe Utilitária

Funções estáticas utilitárias

Uma curiosidade é que as funções utilitárias são nada mais nada menos que a programação estruturada dentro da POO.

Primeira atenção na seu dia a dia e criar classes utilitárias, agrupadas por tema, de forma que você possa compartilhar com todo seu time e assim evitar duplicidade de código.

Como identificar uma função utilitária?

Verifique se ela atende alguns requisitos simples:

1. Ela consegue se resolver por ela mesma, sem dependências externas?
2. Os parâmetros de entrada são simples e diretos?
3. O resultado da saída também é simples e direto?

Um exemplo de funções utilitária

Um exemplo clássico de função utilitária é a validação de CPF ou CNPJ.

✓ Entrada Simples

✓ Saída Simples

✓ Não depende de recursos externos (ex: Banco de dados)

Note que independente da sua complexidade interna, temos uma entrada simples, o CPF, e uma saída simples, verdadeiro ou falso.

```
public static boolean isCPF(String cpf) {  
    // considera-se erro CPFs formados por  
    // uma sequencia de numeros iguais  
    if (cpf.equals("00000000000"))  
        return false;  
    if (cpf.equals("11111111111"))  
        return false;  
    if (cpf.equals("22222222222"))  
        return false;  
    if (cpf.equals("33333333333"))  
        return false;  
    if (cpf.equals("44444444444"))  
        return false;  
    if (cpf.equals("55555555555"))  
        return false;  
    if (cpf.equals("66666666666"))  
        return false;  
    if (cpf.equals("77777777777"))  
        return false;  
    if (cpf.equals("88888888888"))  
        return false;  
    if (cpf.equals("99999999999"))  
        return false;  
    if (cpf.length() != 11)  
        return false;  
    return true;  
}
```

O que é classe e objeto e primeiros passos no código

Conceitos básicos da orientação a objetos

Classe e Objeto

Uma representação dados em objetos ou entidades para o processamento de outros objetos.

Associação de Classes

Quando utilizamos uma classe dentro de outra classe.

Herança

É a utilização de uma classe base, fazendo com que uma nova classe tenha todos os atributos e funções da classe pai, mais a suas próprias.

Encapsulamento

E a possibilidade de proteger alguns dados ou funcionalidades da classe, não permitindo que seus consumidores possam acessá-las.

Polimorfismo

Aqui podemos criar funções que terão o mesmo nome, mas que podem ter diferentes processamentos, implementações ou na mesma classe o mesmo nome e diferentes entradas.

Observação: Tudo que pode começar com 0, colocamos como String (String Numérica. Pois, se não colocarmos, teremos que fazer um tratamento para adicionar o 0 na frente em algumas linguagens de programação.



Moisés Baddini

Outra dica nesse assunto de 'identificar' o que é inteiro ou string: Posso "somar os dois" e obter um novo valor? é inteiro. Não posso fazer operações com eles? é string.