CEUB

EDUCAÇÃO SUPERIOR

Programação Orientada a Objetos

ceub.br

Agenda

Paradigmas de Programação

Paradigma Orientado a Objetos

Introdução a Programação Orientada a Objetos

Pilares da Programação Orientada a Objetos

Elementos básicos da POO (Classes, Objetos, Métodos, Atributos e Instancias)

Orientação a Objetos

Paradigmas de Programação

Quando uma linguagem de programação é criada, a partir das suas características, ela é categorizada em um ou mais paradigmas.

A definição do dicionário Aurélio para "paradigma":

- **Algo que serve de exemplo geral ou de modelo.**
- **Conjunto das formas que servem de modelo de derivação ou de flexão.**
- **Conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.**

Paradigmas de Programação

O paradigma de uma linguagem de programação é a sua identidade. Corresponde a um conjunto de características que, juntas, definem como ela opera e resolve os problemas. Algumas linguagens, inclusive, possuem mais de um paradigma, são as chamadas multi paradigmas, Ex JScript.

Alguns dos principais paradigmas utilizados hoje no mercado:

- **Funcional**
- **Lógico**
- **Declarativo**
- **Imperativo**
- **Orientado a objetos**
- **Orientado a eventos**

Paradigmas de Programação

Paradigma Funcional

Trata a computação como uma avaliação de funções matemáticas e que evita estados ou dados mutáveis. Ela enfatiza a aplicação de funções (Linguagem Haskell).

Exemplo Imperativo

```
/** Média de vendas imperativo - verboso e improdutivo. **/
@Test
public void exer6() {
  List<Fatura> vendas = getVendas();
  double total = 0;
  for (int i = 0; i < vendas.size(); i++) {
    total += vendas.get(i).getValor();
  }
  double media = total / vendas.size();
  System.out.println(media);
}
```

Exemplo Funcional

```
/** Média de vendas funcional - limpa, consisa, rapida, facil. **/
@Test
public void exer7() {
  double total = getVendas().stream().mapToDouble(f -> f.getValor()).average().getAsDouble();
  System.out.println(total);
}
```

Paradigmas de Programação

Paradigma Lógico

Também é conhecido como "restritivo". Muito utilizado em aplicações de inteligência artificial.

Esse paradigma chega no resultado esperado a partir de avaliações lógico-matemáticas.

Principais elementos desse paradigma:

Proposições: base de fatos concretos e conhecidos.
Regras de inferência: definem como deduzir proposições.
Busca: estratégias para controle das inferências.
Exemplo:

Proposição: José é uma pessoa.
Regra de inferência: Todo pessoa é um mamífero .
Busca: Jose é um mamífero?

Paradigmas de Programação

Paradigma Declarativa

O paradigma declarativo é baseado no lógico e funcional.

Linguagens declarativas descrevem o que fazem e não exatamente como suas instruções funcionam.

Linguagens de marcação são o melhor exemplo: HTML, XML, XSLT, XAML etc.

```
<Button x:Name="myActivateLicenseButton" Grid.Column=  15 warnings
        Click="OnActivateLicenseClick" Padding="4,2,    Click to navigate to the next warning
        Margin="2" />

<StackPanel Grid.Column="0" Grid.ColumnSpan="3" Grid.Row="13" HorizontalAlignment
    <Grid Background="Tomato"
          Visibility="{Binding HasErrorOccured, Converter={StaticResource myBoole
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Image Margin="6,3,3,3">
            <Image.Source>
                <Icons:ThemedIconExtension TypeofIcon="{x:Type resources:CommonTh
            </Image.Source>
        </Image>
```

DATA | RESULTS | SQL | Calculations | Row Limit 500 | Totals

```
SELECT
    products.brand  AS "products.brand",
    products.category  AS "products.category",
    COUNT(DISTINCT products.id ) AS "products.count"
FROM public.order_items  AS order_items
LEFT JOIN public.inventory_items  AS inventory_items ON order_items
    .inventory_item_id = inventory_items.id
LEFT JOIN public.products  AS products ON inventory_items.product_id = products.id

GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 500
```

Open in SQL Runner | Explain in SQL Runner

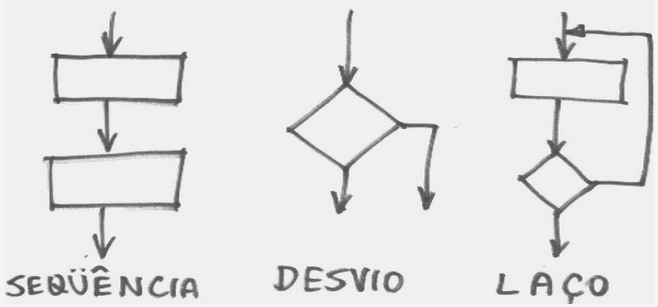
Paradigmas de Programação

Paradigma Imperativo

O paradigma imperativo é aquele baseado em ordens, instruções e comandos, o programador diz como e o quê exatamente um programa ou rotina deve realizar.

É neste paradigma que surgiram os famosos laços de repetição, estruturas condicionais, atribuição de valor à variáveis e controle de estado.

A maioria de nós programadores utilizamos este paradigma de programação no nosso dia a dia sem, muitas vezes, nos darmos conta disto.



Paradigmas de Programação

Paradigma Orientado a Objetos

Esse é, entre todos, talvez o mais difundido. Nesse paradigma, ao invés de construirmos nossos sistemas com um conjunto estrito de procedimentos, assim como se faz em linguagens "fortemente imperativas" como o Cobol, Pascal etc.

Na orientação a objetos utilizamos uma lógica bem próxima do mundo real, lidando com objetos, estruturas que já conhecemos e sobre as quais possuímos uma grande compreensão.

O paradigma orientado a objetos tem uma grande preocupação em esconder o que não é importante e em realçar o que é importante.

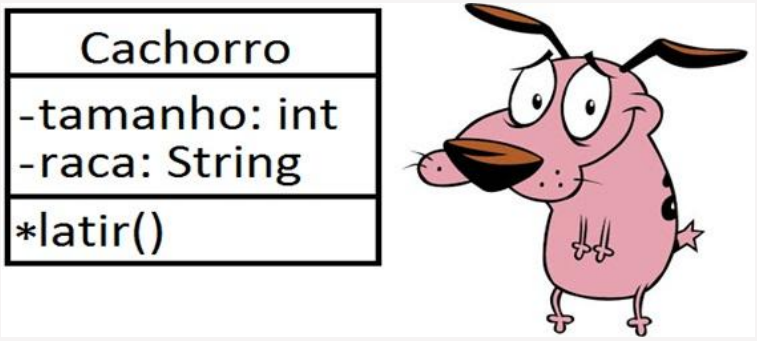
Paradigmas de Programação

Paradigma Orientado a Objetos

Nele, implementa-se um conjunto de classes que definem objetos.

Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento entre eles.

Esse é o paradigma mais utilizado em aplicações comerciais e as principais linguagens o implementam: C#, C#, PHP, Ruby, C++, Python etc.



Introdução a Programação Orientada a Objetos

Conceitos

“Uma nova maneira de pensar os problemas utilizando conceitos do Mundo Real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade” Raumbaugh

“Sistema orientado a objetos é uma coleção de objetos que interagem entre si” Bertrand Meyer

O que é a Orientação a Objetos?

Orientação a Objetos (OO) é um paradigma de programação;

- **Forma diferente de pensar e construir algoritmos e programas para computadores;**

Conceito introduzido originalmente na linguagem de programação Simula 67:

- **Ole-Johan Dahl e Kristen Nygaard;**
- **Oslo, Noruega em 1967;**
- **Nos anos 70 surge Smalltalk, a primeira linguagem totalmente em Orientação a Objeto (O.O)**

  C++, evolução de C, já possuía conceitos O.O

Na década de 80 praticamente todas as linguagens já usavam conceitos O.O

- **Delphi**
- **JAVA**
- **C#(2002)**

Análise, Projeto e Programação OO

- **Na Análise OO o objetivo é desenvolver um modelo OO do domínio da aplicação. Os objetos identificados podem ou não podem ser mapeados diretamente em objetos do sistema.**

- **No Projeto OO o objetivo é desenvolver um modelo OO de um sistema de software para implementar os requisitos identificados. Estes requisitos podem ou não ser estruturados em objetos no domínio do problema.**

- **Na Programação OO o objetivo é realizar um projeto de software usando uma linguagem de programação OO. Uma linguagem de programação OO suporta diretamente a implementação de objetos e fornece classes e herança**

O que é a Orientação a Objetos?

A Programação Orientada a Objetos (POO) é praticada com novos conceitos:

- **Modelagem de Objetos;**
- **Atributos e comportamentos;**

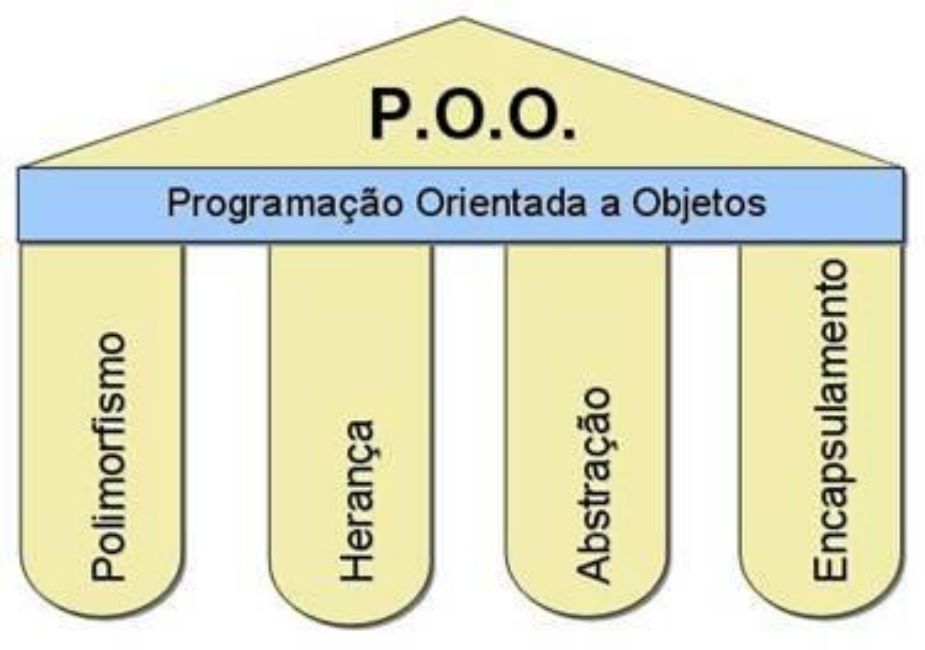
Tudo passa a ser assumido como objetos, tal como acontece no mundo real;

- **Isso trás diferenças substanciais quanto à forma de se pensar na construção de algoritmos!**

Os 4 pilares da Programação Orientada a Objetos

Para que uma linguagem possa ser enquadrada no paradigma de orientação a objetos, ela deve atender a quatro tópicos bastante importantes:

- **Abstração**
- **Encapsulamento**
- **Herança**
- **Polimorfismo**



Os 4 pilares da Programação Orientada a Objetos

Abstração

A abstração consiste em um dos pontos mais importantes dentro de qualquer linguagem Orientada a Objetos.

Como estamos lidando com uma representação de um objeto real (o que dá nome ao paradigma), temos que imaginar o que esse objeto irá realizar dentro de nosso sistema.

Habilidade de se concentrar nos aspectos essenciais do sistema, ou um contexto qualquer, ignorando o que é supérfluo.

Os 4 pilares da Programação Orientada a Objetos

Abstração

Ao observar um objeto podemos descrevê-lo;
Como você descreve o objeto abaixo?

Características?
Funcionalidades?

Os 4 pilares da Programação Orientada a Objetos

Abstração

Na abstração, iremos suprimir (abstrair) as características imutáveis e irrelevantes;

Os 4 pilares da Programação Orientada a Objetos

Abstração

Características:
Cor: Vermelho
Modelo: MODEL 3
Marca: Tesla
Motorização: Elétrica

Funcionalidades

- Acelerar
- Frear
- Buzinar
- Retroceder

Os 4 pilares da Programação Orientada a Objetos

Encapsulamento

O *encapsulamento* é uma das principais técnicas que define a programação orientada a objetos.

Se trata de um dos elementos que adicionam segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta.

Essa atitude evita o acesso direto a propriedade do objeto, adicionando uma outra camada de segurança à aplicação.

Os 4 pilares da Programação Orientada a Objetos

Encapsulamento

- Para fazermos um paralelo com o que vemos no mundo real, temos o encapsulamento em outros elementos.
- Por exemplo, quando clicamos no botão ligar da televisão, não sabemos o que está acontecendo internamente.
- Podemos então dizer que os métodos que ligam a televisão estão encapsulados.

Os 4 pilares da Programação Orientada a Objetos

Herança

O reuso de código é uma das grandes vantagens da programação orientada a objetos. Muito disso se dá por uma questão que é conhecida como *herança*.

Essa característica otimiza a produção da aplicação em tempo e linhas de código.

A reutilização de códigos nas linguagens orientadas a objetos é uma característica que otimiza o desenvolvimento de um aplicativo tanto em economia de tempo, quanto em número de linhas de código.

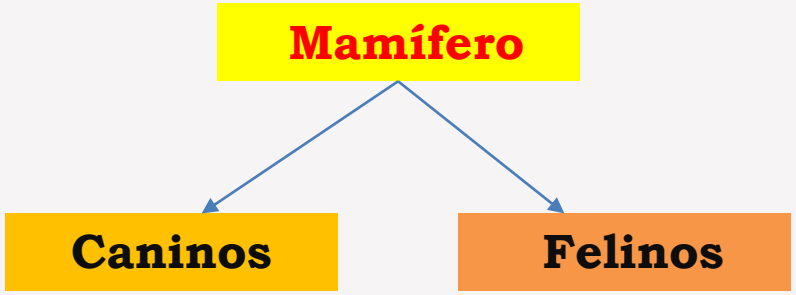
Os 4 pilares da Programação Orientada a Objetos

Herança

Por exemplo, analisemos uma classe com todas as características dos animais mamíferos.

Se tivermos uma classe Humanos e outra felinos ambas as classes herdarão as características da classe mamíferos.

Do ponto de vista de código, ao criar as classe estaremos economizando códigos e tempo de desenvolvimento.



Os 4 pilares da Programação Orientada a Objetos

Polimorfismo

O quarto e último pilar da POO é conhecido como polimorfismo. Trata-se da capacidade de alterar a forma original conforme a necessidade do momento.

A palavra polimorfismo vem do grego e significa muitas formas (poli: muitas, morphos: formas);

Em suma, o polimorfismo consiste na alteração de todo o funcionamento interno de um método herdado de um outro objeto dentro da aplicação.

Principais vantagens da POO

Flexibilidade

- Um código flexível quanto a mudanças é muito desejável, e é justamente o que se alcança ao se aplicar bem os conceitos da orientação a objetos.

Reusabilidade

- É um dos principais requisitos no desenvolvimento de softwares. Como os sistemas estão cada vez mais complexos, o tempo para desenvolve-los de forma assertiva ficaria cada vez maior se não fosse possível a reutilização.

Principais vantagens da POO

Robustez

- Ao se usar uma linguagem orientada a objetos é possível criar sistemas robustos, confiáveis, extensíveis, reutilizáveis e mantenéveis.

Modularidade

- Modularização é o processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente.

- Particionar um programa em componentes modulares, individuais, pode reduzir a complexidade.

Programação Orientada a Objetos

Elementos básicos

Objetos

Classes

Instâncias

Programação Orientada a Objetos

Elementos básicos - OBJETOS

Objetos são entidades concretas ou abstratas

- **Tem características e podem executar ações**
- **"Um objeto representa um item identificável, uma unidade ou entidade, individual, seja real ou abstrato, com uma regra bem definida" com uma regra bem definida"**
- **Possuem:**
  - **Estado**
  - **Comportamento (MÉTODOS)**
  - **Identidade**

Programação Orientada a Objetos

Estado do Objeto

- **Define os estados possíveis que um objeto pode assumir.**
- **São os valores dos atributos (propriedades)**
- **Ex : Lâmpada dois estados:**

Acesa

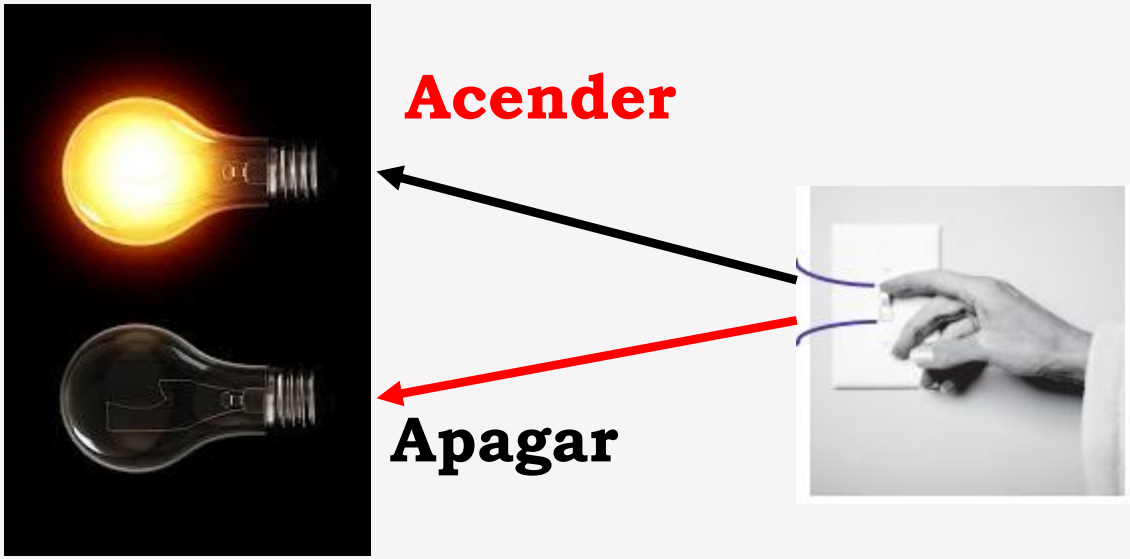
Apagada

Programação Orientada a Objetos

Comportamento do Objeto

São as funções que podem ser executadas por um determinado objeto;

- **Corresponde aos métodos**
- **O que você pode fazer com esse objeto**



Programação Orientada a Objetos

Identidade do Objeto

- **Um objeto é único, mesmo que o seu estado seja idêntico ao de outro**

CodLampada =1002

CodLampada =1003

Programação Orientada a Objetos

CLASSE

Modelo ou esquema a partir do qual os objetos são criados (instanciados).

Modelam os objetos definindo:

- **Tipo de dados que o objeto armazena, ou seja, os estados possíveis que ele pode assumir (atributos)**
- **Tipos de operações que podem ser executadas pelo objeto, ou seja, o seu comportamento (métodos)**

Abstração de objetos de características semelhantes (molde).É a essência do objeto.

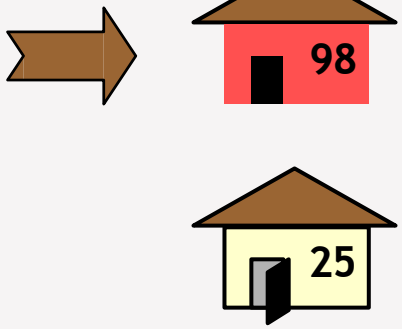
Objetos são instâncias de classes.

O que é uma classe?

- *Classes são uma especificação para objetos*
- *Uma classe representa um tipo de dados (é uma estrutura de dados) complexo*
- *Classes descrevem*
  - *Tipos dos dados que compõem o objeto (o que podem armazenar)*
  - *Procedimentos que o objeto pode executar (o que podem fazer)*

Instâncias da classe Casa (objetos)

Classe




```
Casa c1 = new Casa();
c1.numero = 1;
c1.cor = Color.yellow;

Casa c2 = new Casa();
c2.numero = 98;
c2.cor = Color.red;

Casa c3 = new Casa();
c3.numero = 25;
c3.cor = Color.white;
c3.abrePorta();
```

O que é uma classe?

- ***Classes não são os objetos que representam***
  - ***A planta de uma casa é um objeto, mas não é uma casa***
- ***Classes definem lógica estática***
  - ***Relacionamentos entre classes são definidos na programação e não mudam durante a execução***
  - ***Relacionamentos entre objetos são dinâmicos e podem mudar. O funcionamento da aplicação reflete a lógica de relacionamento entre os objetos, e não entre as classes.***
- ***Classes não existem no contexto da execução***
  - ***Uma classe representa vários objetos que ocupam espaço na memória, mas ela não existe nesse domínio***
  - ***A classe tem papel na criação dos objetos, mas não existe quando os objetos trocam mensagens entre si.***

INSTANCIA

INSTANCIA É SINONIMO DE UM OBJETO;

O ATO DE INSTANCIAR UMA CLASSE É O ATO DE CRIAR UM OBJETO.

Em C#, uma instância de uma classe é um novo objeto criado dessa classe, com o operador new.

Instanciar uma classe é criar um novo objeto do mesmo tipo dessa classe.

Uma classe somente poderá ser utilizada após ser instanciada.

CLASSES X OBJETOS

• Uma classe é um modelo ou protótipo que define as propriedades e métodos (comportamento) comuns a um conjunto de objetos;

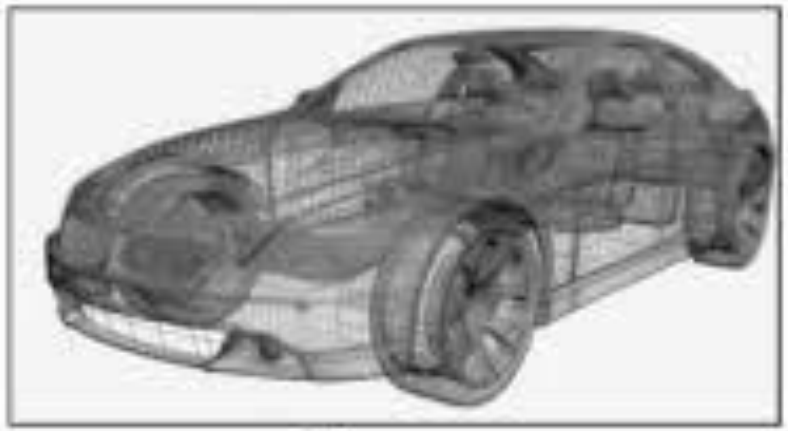
• Classes são “moldes” que definem as variáveis e os métodos comuns a todos os objetos de um determinado tipo;

• No mundo real existem vários objetos do mesmo tipo. Por exemplo, o seu carro é um dos milhares que existem no mundo;

• Usando a terminologia de orientação a objetos, o objeto seu carro é uma instância da classe de objetos carro.

CLASSES X OBJETOS

- O grupo de objetos que possuem os mesmos atributos e métodos diz-se que pertencem à mesma classe.



Classe

Objeto

CLASSES

Sintaxe básica de criação de classes em C#

Uma classe em C# é sempre declarada com a palavra-chave class seguida do nome da classe.

O nome da classe não pode conter espaços, deve sempre ser iniciado por uma letra.

Para nomes de classes, métodos e campos em C#, o caractere sublinhado (_) e o sinal $ são considerados letras.

O nome da classe não deve conter acentos1
e pode conter números, contanto que estes apareçam depois de uma letra.

CLASSES

Sintaxe básica de criação de classes em C#

Nomes de classes não podem ser exatamente iguais às palavras reservadas de C#.

• Caracteres maiúsculos e minúsculos são diferenciados em C#: as palavras Class, CLASS, ClAsS e class são consideradas como sendo diferentes, e somente a última pode ser usada para declarar uma classe.

Ex

```
public class NomeClasse {

}
```

```
class NomeClasse {

}
```

CLASSES

Sintaxe básica de criação de objetos em C#

Para criar (construir, instanciar) uma classe, basta usar a palavra chave new.
Ex

```
using System;

namespace ExemploOOClasses
class NomeClasse {
 public static void main(String[] args) {
      NomeClasse meuObjeto;
      meuObjeto = new NomeClasse();

      ou

      NomeClasse meuObjeto = new NomeClasse();

  }
}
```

Referências

https://www.treinaweb.com.br/blog/linguagens-e-paradigmas-de-programacao/

https://www.devmedia.com.br/vantagens-e-desvantagens-da-poo/32655