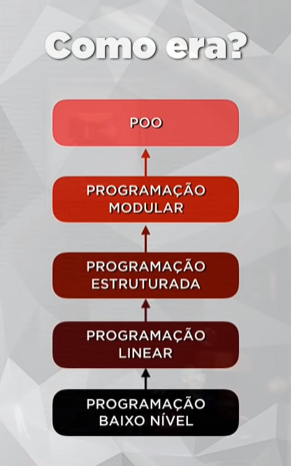
**AULA TEÓRICA 1 – O QUE É PROGRAMAÇÃO ORIENTADA A OBJETOS?**

**OBJETIVO DA POO**

Aproximar o mundo digital do mundo real.

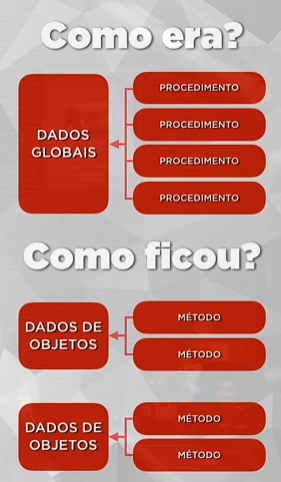
**COMO ERA?**



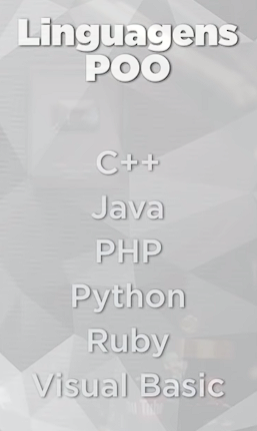
**QUEM CRIOU?**



**COMO FICOU?**

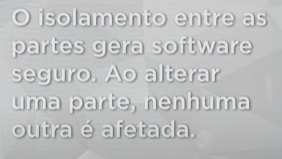


**LINGUAGENS POO**

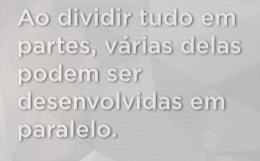


**QUAIS AS VANTAGENS DA POO?**

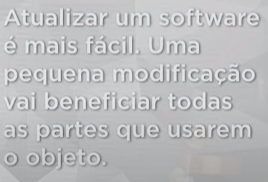
Confiável:



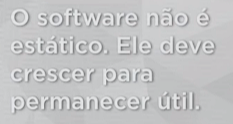
Oportuno:



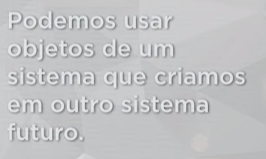
Manutenível:



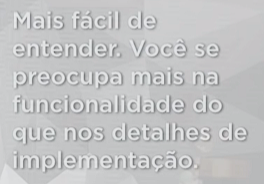
Extensível:



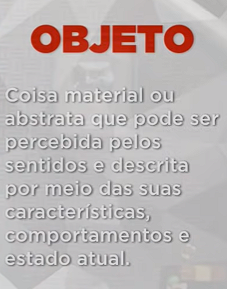
Reutilizável:



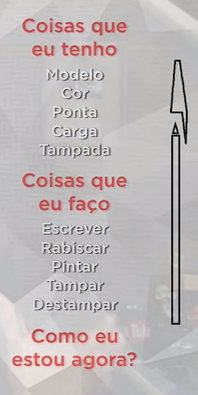
Natural:



# **Aula Teórica 2 – O que é um Objeto?**

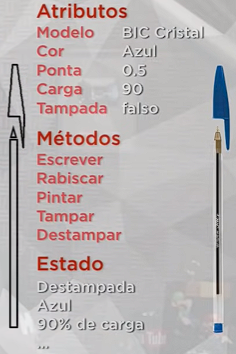


**TODO OBJETO VEM DE UMA CLASSE:**

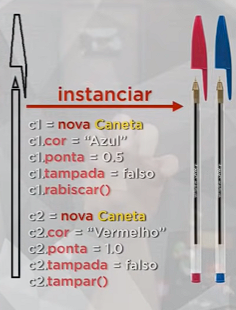


**TODO OBJETO TEM:**

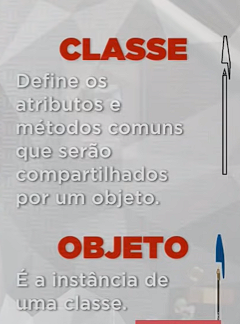




**PARA CRIAR UMA OBJETO É PRECISO INSTANCIAR UMA CLASSE:**

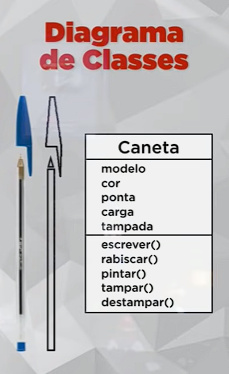


**DEFINIÇÃO TEÓRICA**

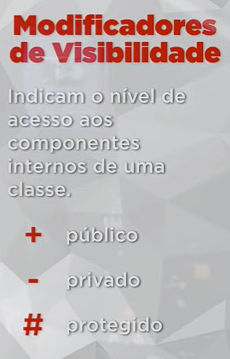


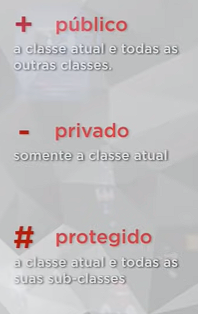
# **Aula Teórica 3 – O que é Visibilidade em um Objeto?**

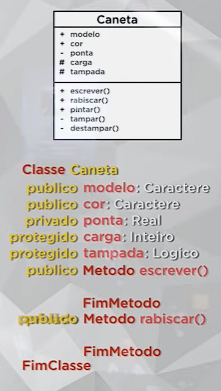
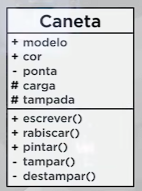
UML - Diagrama de Classes



Modificadores de Visibilidade







# **Aula Teórica 4 – Métodos Especiais**

Método Assessores = Getters

Cada atributo da classe recebe um método get

Métodos Modificadores = Setters

Cada atributo da classe recebe um método set

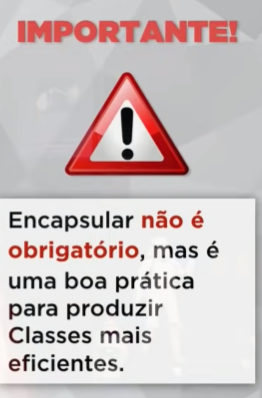
Métodos Construtores = Construct

Define os valores default (atributos e métodos) do objeto quando uma classe é instanciada.

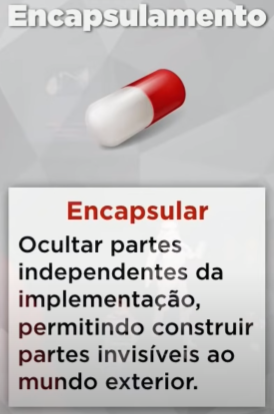
# **Aula Teórica 6 – Pilares da POO: Encapsulamento**

A programação orientada a objetos possui três pilares:

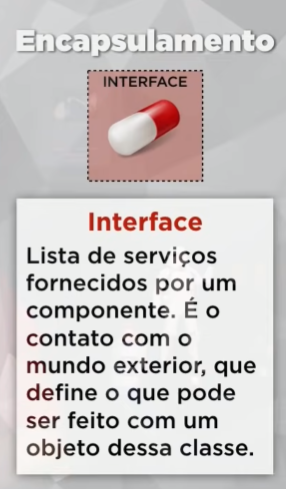
* Encapsulamento
  + Em algumas bibliografias Abstração também é um pilar
* Herança
* Polimorfismo
* **IMPORTANTE!!!**

****

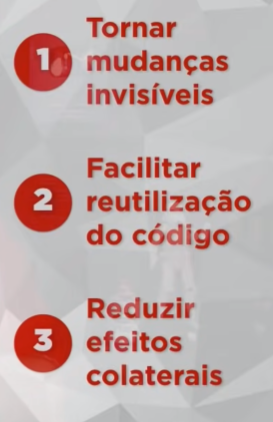
* **Encapsulamento:**



* **Interface:**

****

* **Vantagens de encapsular:**

****

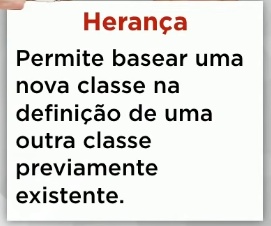
# **Aula Teórica 7 – Relacionamento entre Classes**

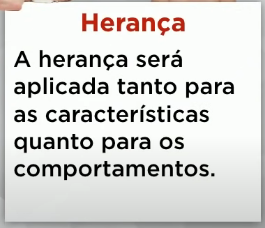
# **Aula Teórica 7a – Objetos Compostos**

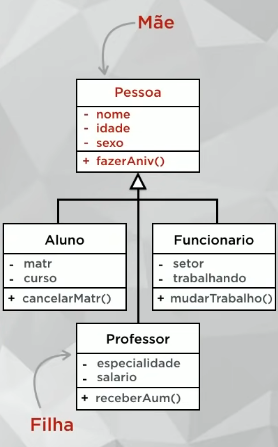
# **Aula Teórica 8a – Relacionamento de Agregação**

# **Aula Teórica 8b – Agregação entre Objetos com Java**

# **Aula Teórica 10a – Herança**



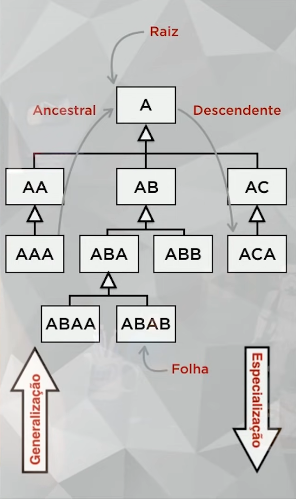


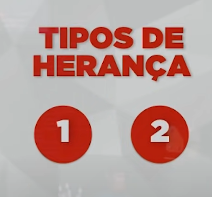


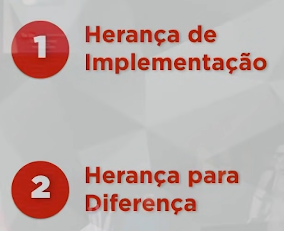
# **Aula Teórica 10b – Herança**

Nessa aula de POO, vamos aprender a aplicar o conceito de Herança em Java.

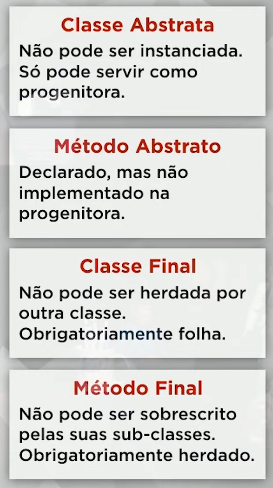
# **Aula Teórica 11a – Herança**





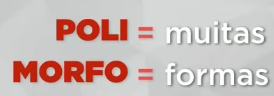


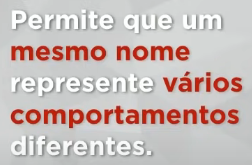


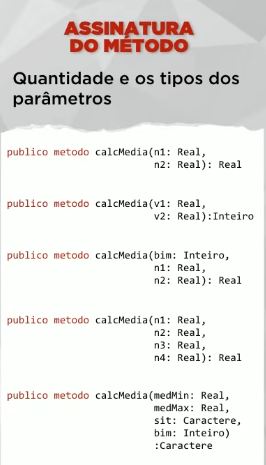


# **Aula Teórica 11b – Herança**

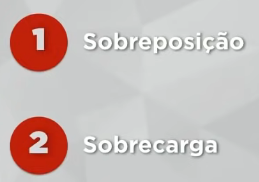
# **Aula Teórica 12a – Conceito Polimorfismo**

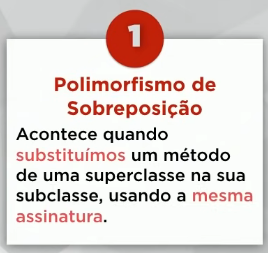












# **Aula Teórica 12b – Conceito Polimorfismo**

# **Aula Teórica 13a – Conceito Polimorfismo**

Sobreposição =

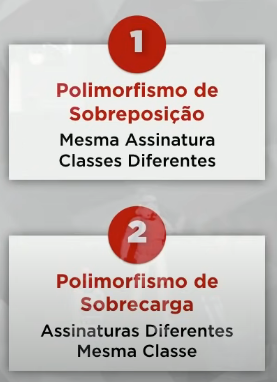
Assinatura igual && Classes diferente:

Quantidade e tipos iguais (parâmetros do método).

Sobrecarga =

Assinaturas diferentes && Mesma Classe:

Quantidades e tipos diferentes (parâmetros do método).



# **Aula Teórica 13b – Conceito Polimorfismo**