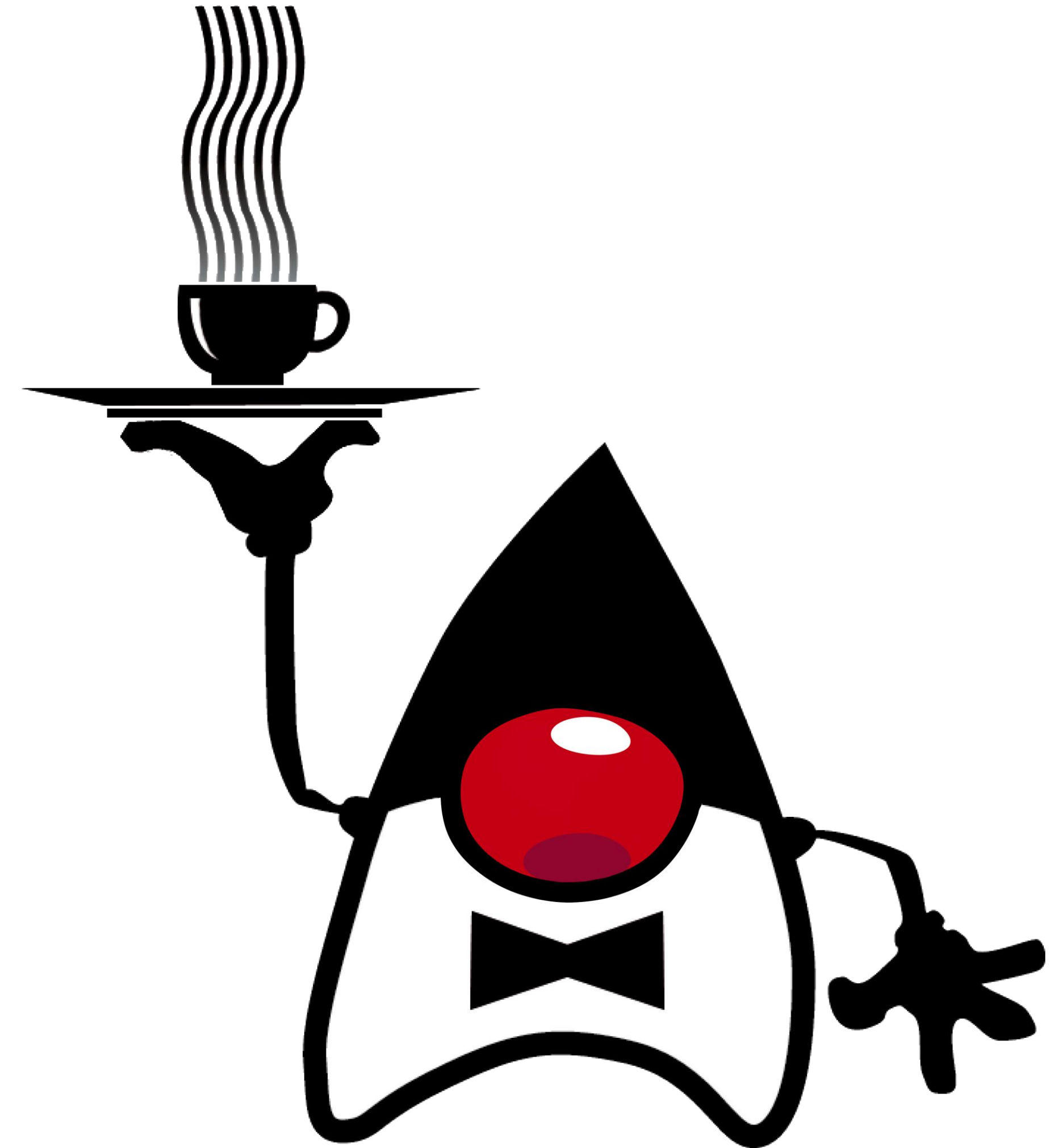


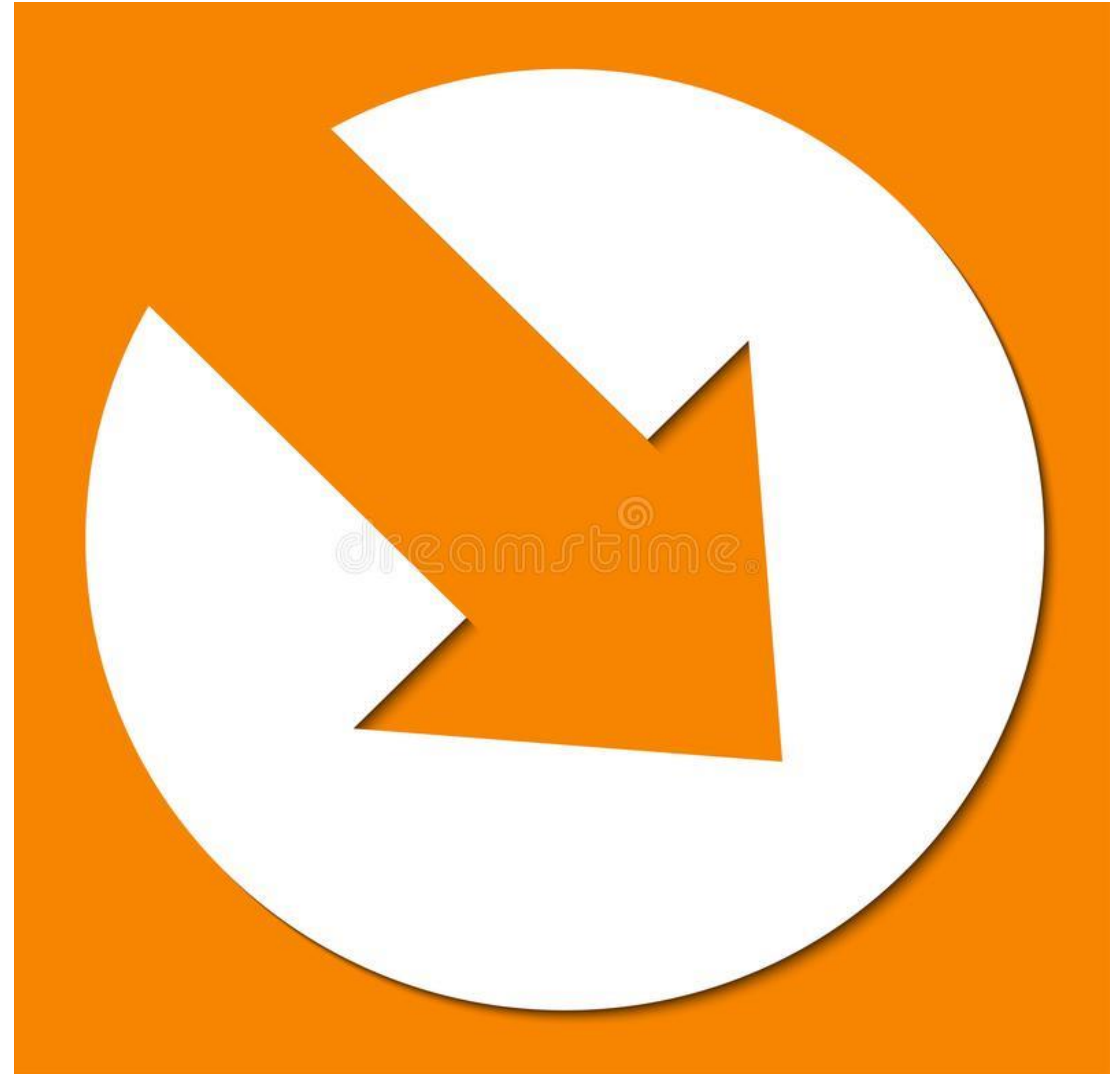
# Trilha Java

Encontro 13 – (POO)  
Criando Classes e Objetos



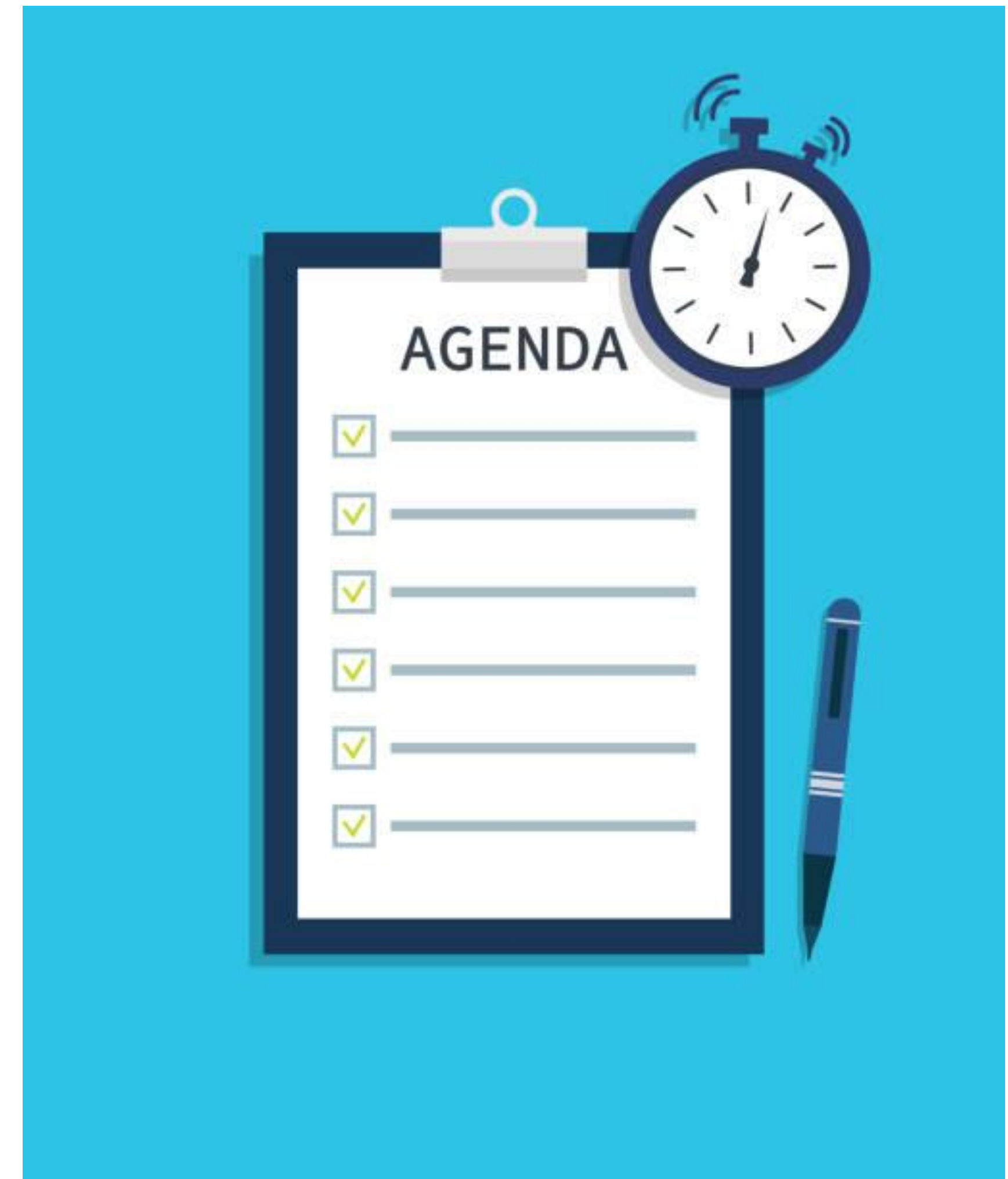
# Recapitulação

1. Vetores
2. Matrizes
3. Métodos(Funções)
4. Exemplos
5. Exercícios



# Agenda

1. Introdução
2. Objetos
3. Classes
4. Criando classes e objetos
5. Abstração
6. Exercícios



# Introdução

## Programação Orientada a Objeto - POO





# POO

O que é Programação Orientação a Objetos?

Projeto e programação de software baseado na composição e **interação** entre diversas unidades, chamadas de '**objetos**'



# POO

Principais características  
da programação orientada  
por objetos

Segurança da informação.

Reaproveitamento de código.



# POO

A programação orientada a objetos é baseada em quatro pilares .

Todos estes conceitos, igualmente importantes, são construídos em cima desses dois: **classes** e **objetos**.





# POO - Abstração

Imagine um objeto e o que ele deve realizar.

**Características** do objeto. Por exemplo, as **propriedades** de um objeto “pessoa” poderiam ser “peso”, “tamanho” e “idade”.

**Ações** que o objeto irá executar, chamadas de **métodos**. Eles podem ser muito variados, dependendo do tipo de solução desenvolvida.



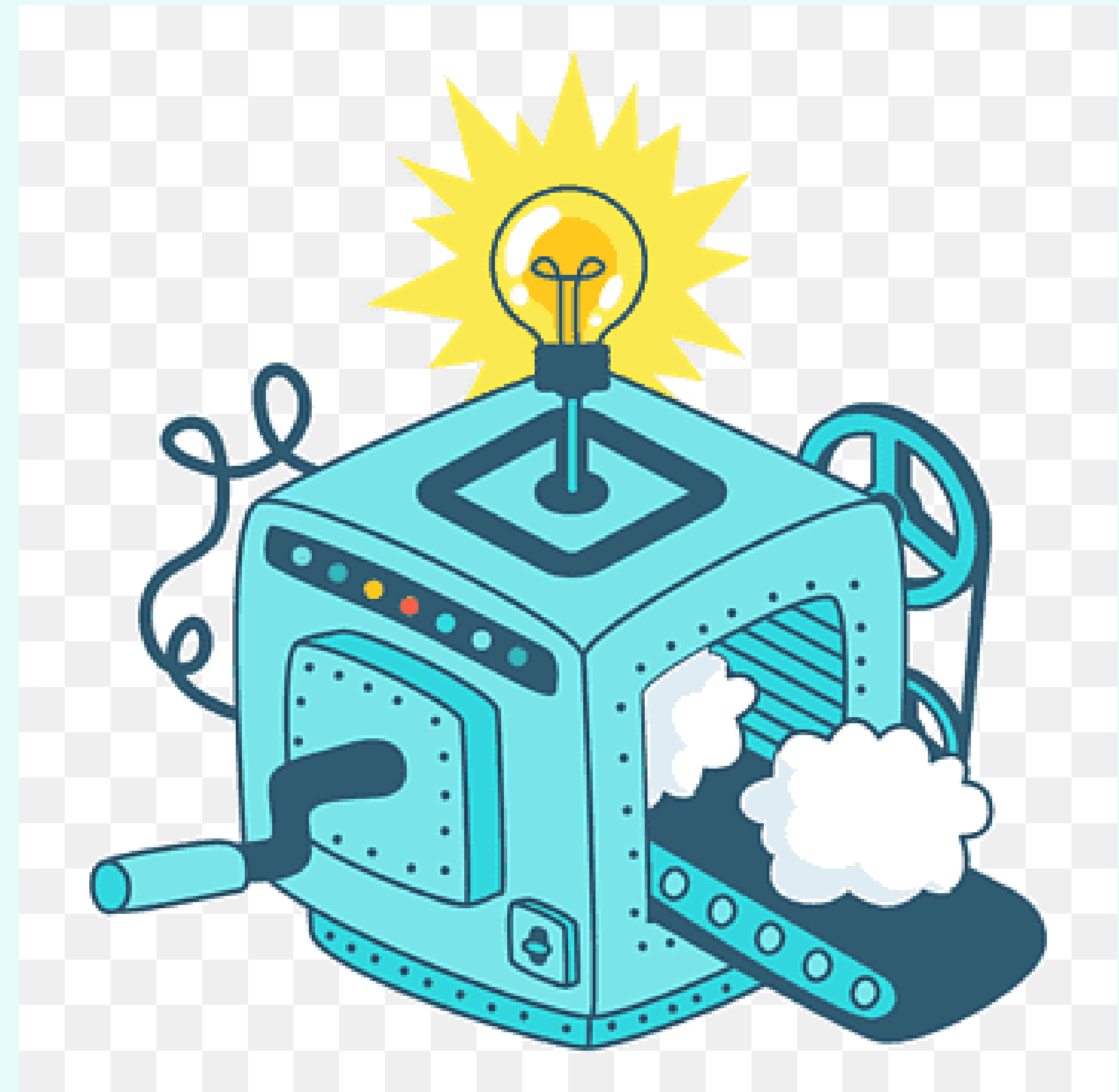


# POO - Encapsulamento

Técnica que adiciona segurança ao software, pois esconde as propriedades, criando uma espécie de caixa preta.

Implementam o encapsulamento baseado em propriedades privadas, por métodos chamados getters e setters.

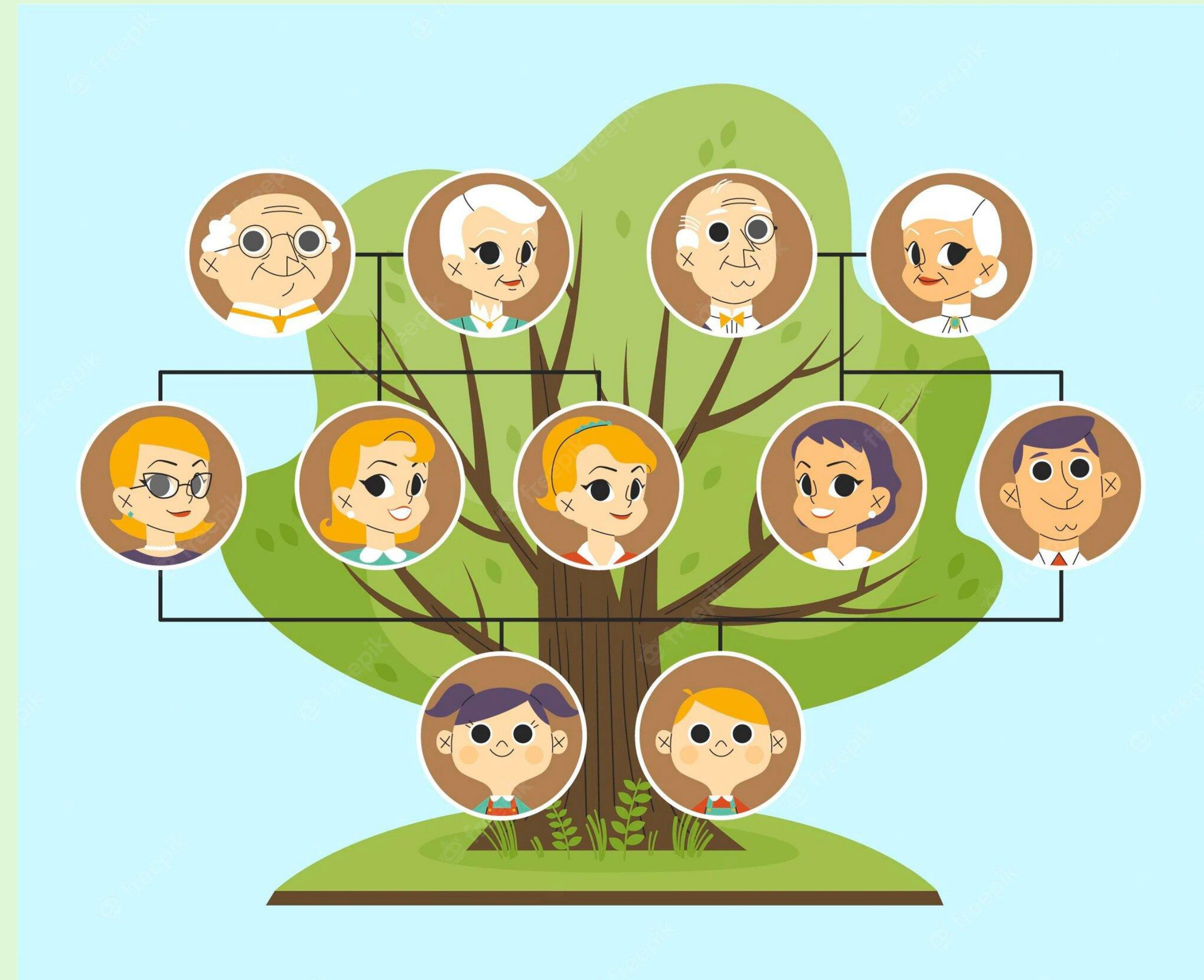
Evita o acesso direto à propriedade do objeto, adicionando outra camada de segurança à aplicação.



# POO - Herança

O **reuso de código** é uma de suas vantagens de destaque e ela se dá por **herança**. Essa característica otimiza a produção do código.

Numa família, por exemplo, a criança herda diretamente do pai e indiretamente do avô e do bisavô. Em programação, a lógica é similar. Os **objetos filhos herdam as características e ações de seus ancestrais**".





# POO - Polimorfismo

Na natureza, existem animais que são capazes de **alterar** sua **forma** conforme a necessidade. Na POO a ideia é a mesma.

O **polimorfismo** permite herdar um método de classe pai e atribuir uma **nova implementação** para o método pré-definido.





# POO - Benefícios

Propõe uma representação mais fácil de ser compreendida e realista.

Reutilização de código.

Otimização do tempo de desenvolvimento.

Facilidade na leitura e manutenção de código.





# Objetos

## Programação Orientada a Objeto



# Objetos

O que é um Objeto?

O que é uma Classe?

Como relacionar tais conceitos?

# Objetos

Possivelmente você reconhece como objeto.



Seria um objeto?

- Horário marcado pra ir ao médico
- Estudante
- Uma venda



# Objetos

Coisa **material** ou **abstrata** que pode ser percebida pelos sentidos e descrita por meio das suas **características**, **comportamentos** e **estado atual**.





# Objetos



## Características

Tem vários botões

Modelo Philco

Plástico

## Comportamentos

Ligar

desligar

Pausar

Gravar

## Estado

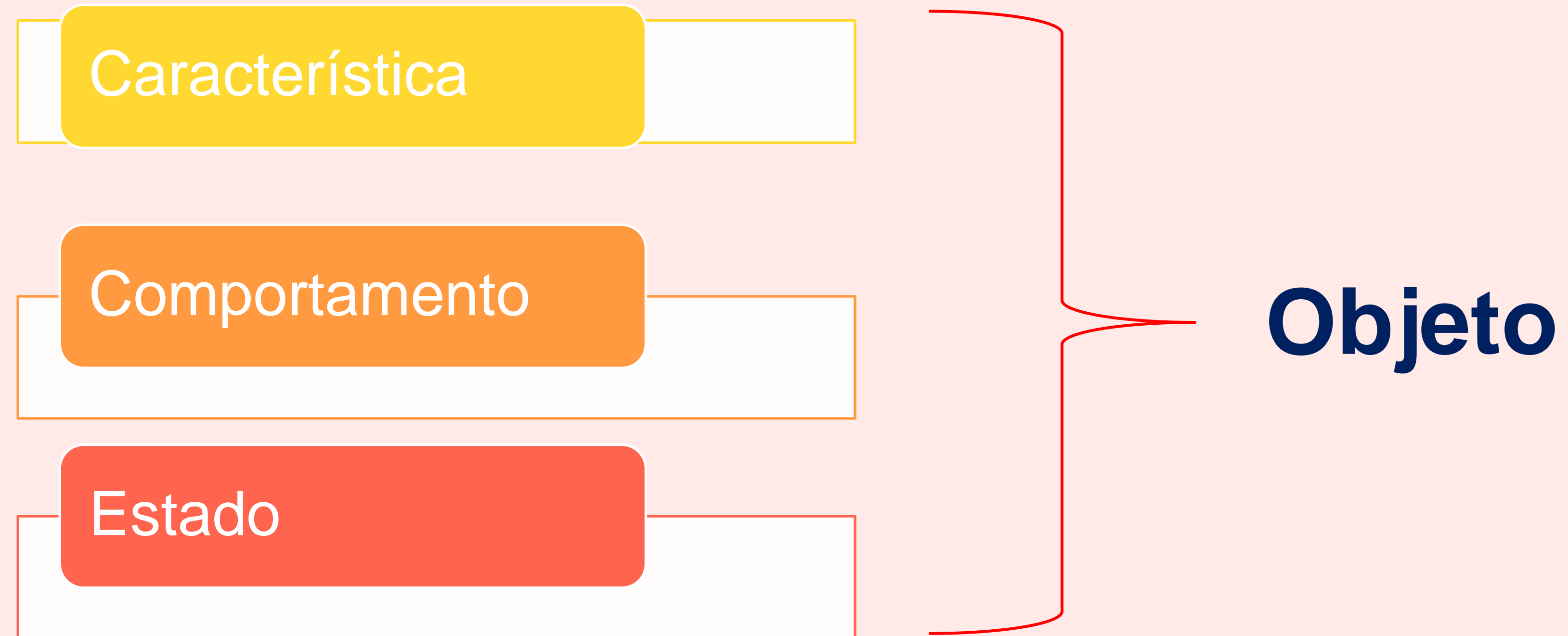
Novo

Sem bateria

desligado

# Objetos

Portanto,



# Objetos

## Compromisso É um objeto?

### Características

- Dia que foi marcado.
- Que tipo de roupa tem que ir.

### Comportamentos

- Pode ser desmarcado
- Pode ser adiado.
- Pode ser cancelado

### Estado Atual

- Está combinado.
- Foi suspenso.

# Objetos

Caneta é um objeto!!

Mas será que só existe  
um objeto caneta?





# Objetos

Existem vários objetos caneta.

Possui:  
mesma forma,  
mesmo formato,  
mesma classificação.

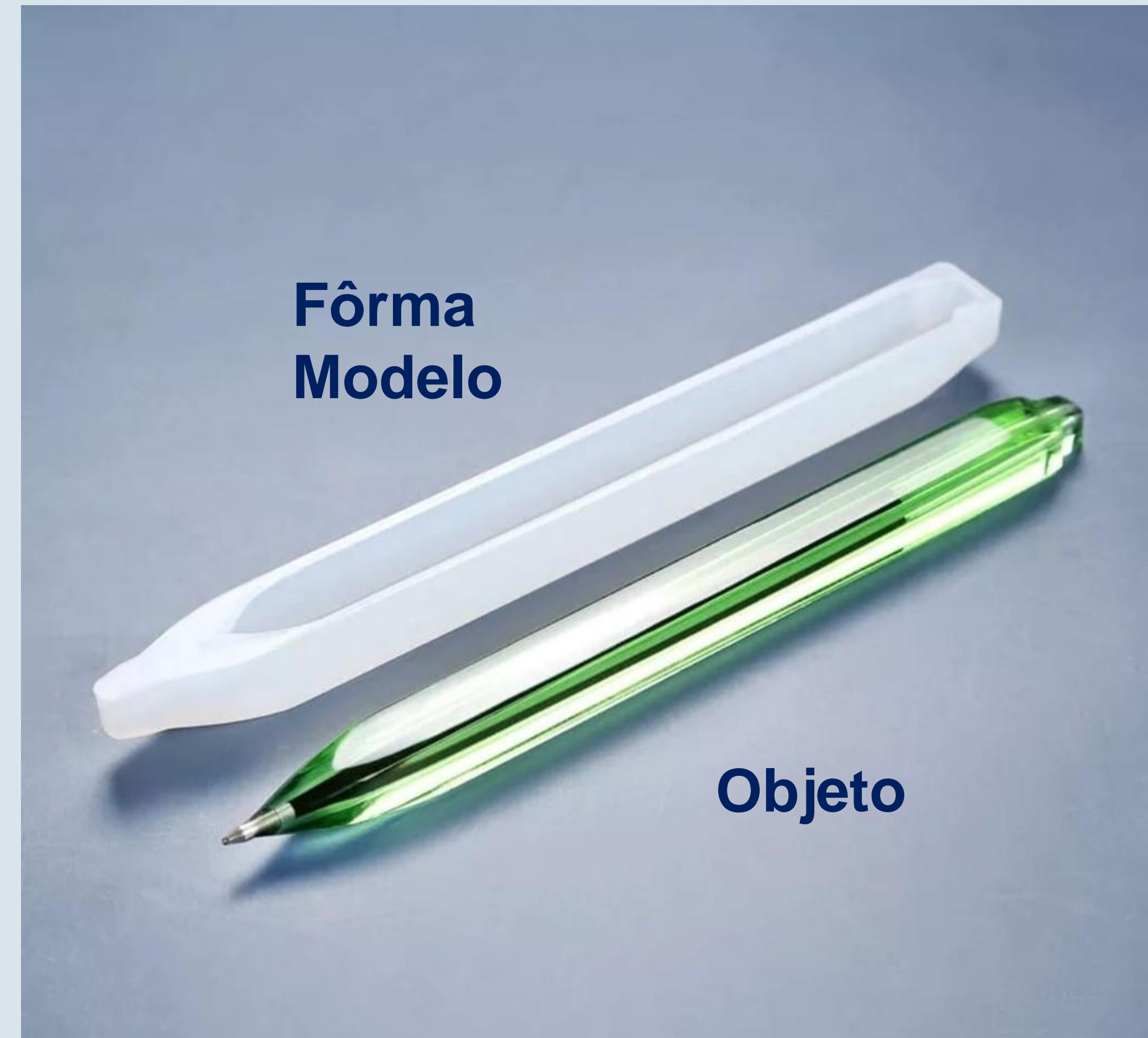


# Objetos

A fôrma ou modelo serve pra produzir os objetos.

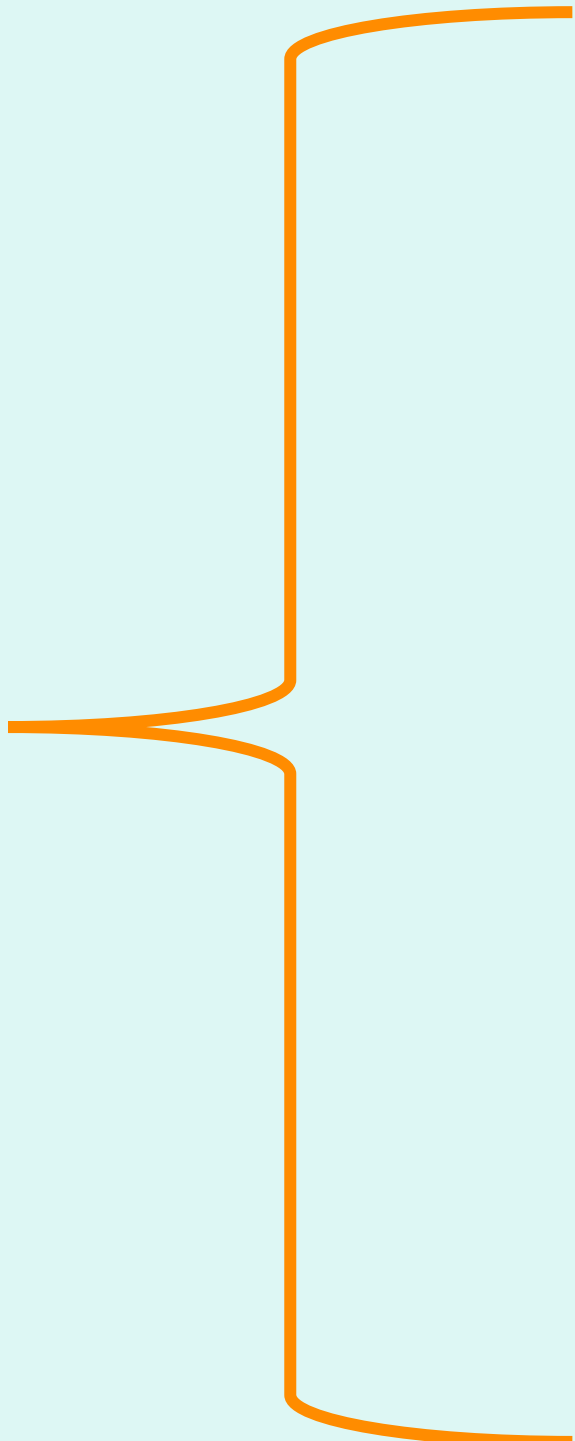
## Em POO:

A fôrma é conhecida como **CLASSE**.  
A caneta em si é um **OBJETO**.



# Objetos

Em POO, uma **CLASSE** tem que atender a três perguntas.



Características que possuo?  
**= Atributos.**

Coisas que eu faço  
**= Métodos.**

Como eu estou agora  
**= Estado**



# Objetos

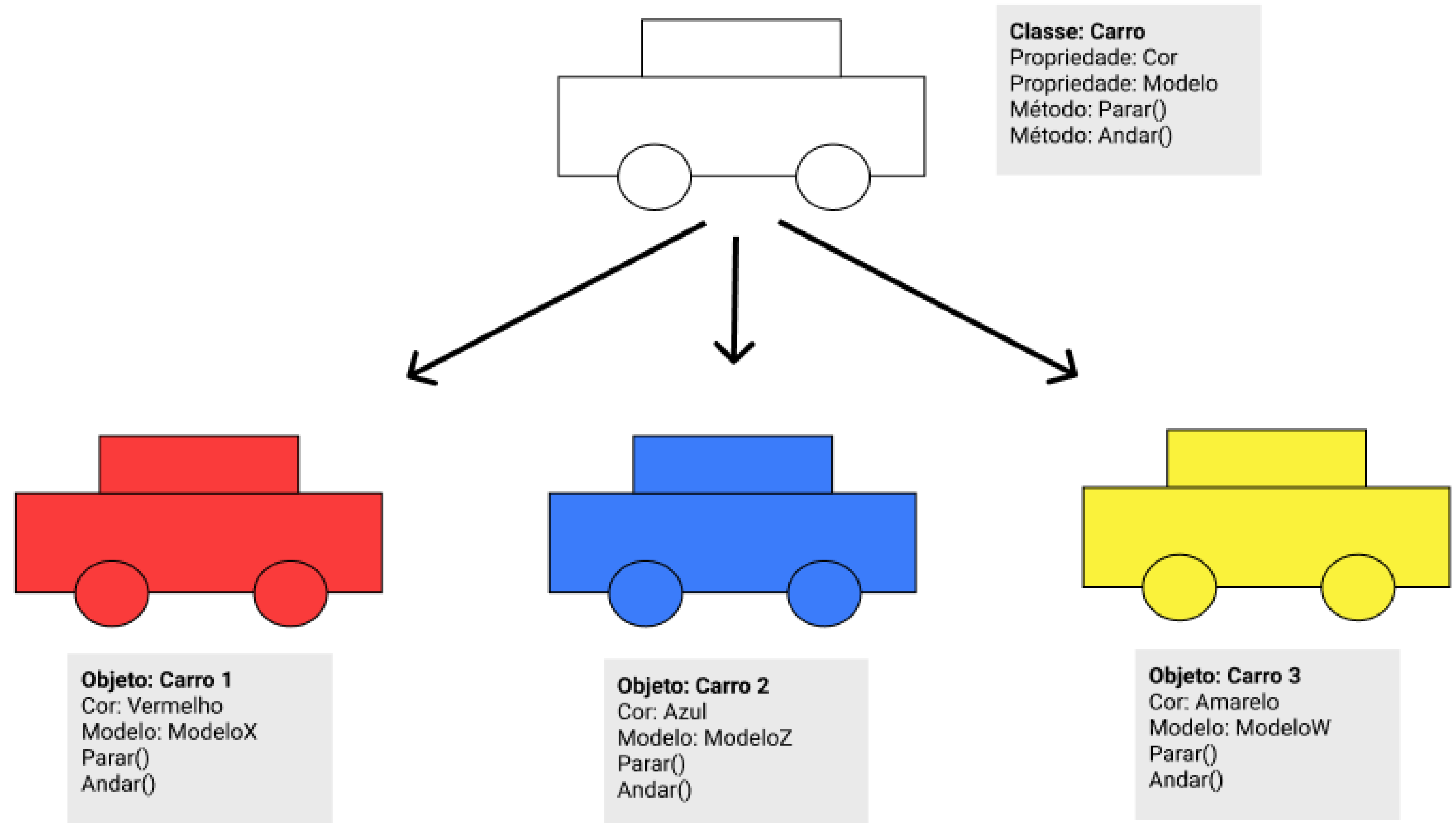
Quem devemos  
criar primeiro, a  
classe ou o objeto?

- Para se criar um objeto, inicialmente se cria o modelo ou a fôrma.
- Portanto, a classe vem antes do objeto



# Objetos

Um outro exemplo:  
Uma **classe** carro  
que produz **objetos**  
carros.



# Classes

## Programação Orientada a Objeto



# Classes

**Exemplo 1: crie  
uma classe caneta.**

Atributos ?  
Métodos ?  
Estados ?

modelo      ponta  
cor      carga  
modelo      tampada  
Escrever      destampada  
tampar      pintar  
rabiscar

# Classes

( ) importante na escrita do método.

## Instanciar:

Quando temos uma classe e geramos através dela um objeto.

## Classe **Caneta**

Modelo: caractere

Cor: caractere

Ponta: real

Carga: inteiro

Tampada: logico

Método rabiscar()

    Se (tampada) então

        Escrever("ERRO")

    Senão

        Escrever("Rabisco")

    Fimse

FimMétodo

Método tampar()

    Tampada = verdadeiro

FimMétodo

**FimClasse**



# Classes

## Temos um objeto:

Quando mencionamos exatamente cada característica, comportamento e estado.

### Atributos

Modelo: **Bic**

Cor: **Azul**

Ponta: **0.7**

Carga: **90%**

Tampada: **falso**

### Métodos

Método rabiscar()

Método tampar()

Métodos pintar()

Método tampar()

### Estado

**Destampada**

**Azul**

**90% de carga**

# Classes

Instanciando um objeto!

**c1** -> Objeto1

**c2** -> Objeto2

```
c1 = new Caneta  
c1.cor = "Azul"  
c1.ponta = 0.5  
c1.tampada = falso  
c1.rabiscar()
```

```
c2 = new Caneta  
c2.cor = "Vermelho"  
c2.ponta = 0.7  
c2.tampada= false  
c2.tampar()
```

# Classes

Portanto,

- **CLASSE:** define os atributos e métodos comuns que serão compartilhados por um objeto.
- **OBJETO:** é a instancia de uma classe.



# Criando Classes e Objetos

Programação Orientada a Objeto



# Criando Classes e Objetos

O primeiro passo é criar a classe principal.

Onde os objetos serão instanciados.

Em seguida crie a classe desejada.

Neste caso iremos criar a classe **Caneta** como exemplo.



# Criando Classes e Objetos

```
package aula13;  
public class Exemplo1 {  
    public static void main(String[] args) {  
        // Aqui devemos instanciar os objetos!!  
    }  
}
```



# Criando Classes e Objetos

```
package aula13;
public class Caneta {
    String modelo, cor;
    float ponta;
    int carga;
    boolean tampada;

    void status() {
        System.out.println("Modelo: " + this.modelo);
        System.out.println("Cor: " + this.cor);
        System.out.println("Tamanho da ponta: " + this.ponta);
        System.out.println("% da carga: " + this.carga);
        System.out.println("Esta tampada? " + this.tampada);
    }

    void rabiscar() {
        if ( tampada == true) {
            System.out.println("Sorry!! Nao posso rabiscar.");
        } else {
            System.out.println("Estou RABISCANDO!!");
        }
    }
}
```

??

# Criando Classes e Objetos

Após a classe **Caneta** ser criada, é possível instanciar vários **objetos caneta**.





# Criando Classes e Objetos

```
package aula13;
public class Exemplo1 {
    public static void main(String[] args) {
        Caneta c1 = new Caneta();
        c1.cor = "Azul";
        c1.modelo = "Bic";
        c1.carga = 90;
        c1.ponta = 0.7f;
        c1.tampada = false;
        c1.rabiscar();
        c1.status();
        System.out.println(" ");
    }
}
```



# Exercícios

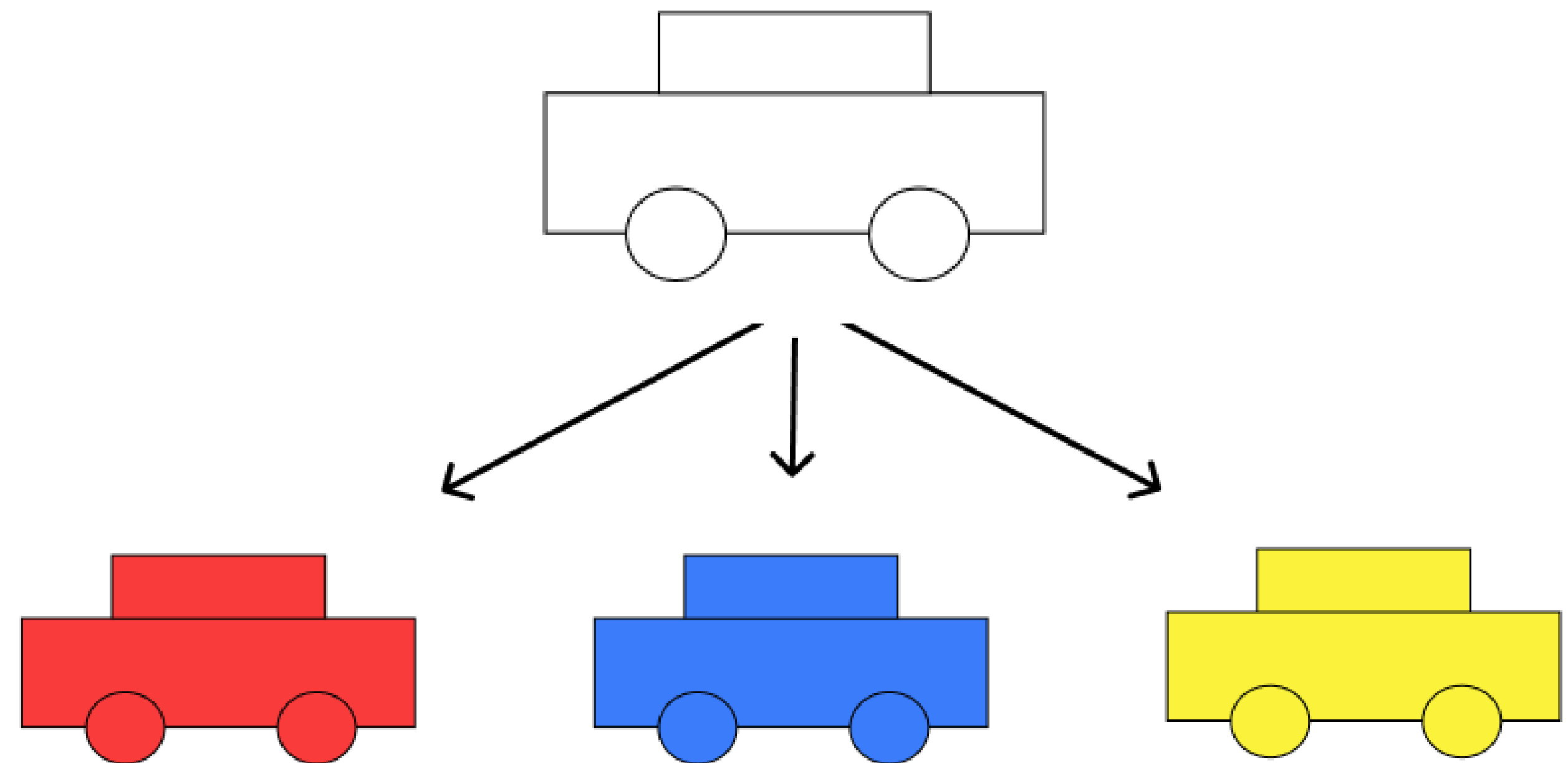
## Programação Orientada a Objeto



# Exercícios

**Atividade 1:** Faça um programa que crie uma classe Carro e em seguida instancie 3 objetos carro1, carro2 e carro3. A classe Carro possui os atributos cor, modelo e velocidade. Além disso, possui os métodos Acelerar, Frear e AcenderFarol. Considere que a velocidade ideal do carro está no seguinte limite.

$$20 \leq \text{Velocidade} \leq 60$$



# Exercícios

**Faça as seguintes considerações:**

- A) Caso a velocidade esteja abaixo do limite, some à velocidade mais 10km/h e imprima uma mensagem "Acelerando!!".
- B) Caso a velocidade esteja acima do limite, subtraia à velocidade o valor de 10km/h e imprima a mensagem "Freando".
- C) Por fim, leve em consideração que o carro parado deve estar com o farol apagado e o carro em movimento deve estar com farol aceso.



# Exercícios

```
package aula13;

public class Atividade1 {
    public static void main(String[] args) {
        Carro carro1 = new Carro();
        carro1.modelo = "Pegeout";
        carro1.cor = "Branco";
        carro1.velocidade = 40.0;
        carro1.status();
        carro1.frear();
        carro1.acelerar();
        carro1.acenderFarol();
        System.out.println(" ");
    }
}
```



Coffee  
time!



# Abstração

## Programação Orientada a Objeto





# Abstração

Suponha que você deseja criar uma **classe cachorro** para estrelar uma campanha de ração.

Ao fazer o levantamento dos **atributos e métodos** da **classe cachorro**, você escolhe apenas quatro atributos e quatro métodos para simplificar.





# Abstração

O que fizemos neste caso foi justamente uma abstração, onde **ocultamos características desnecessárias** do cachorro que não agregam nada ao exemplo.

Imagina que dependendo do objetivo, os atributos e métodos podem ser diferentes.





# Abstração

Classe cachorro para ser guarda-policia.

Classe cachorro para viver em apartamento.

Perceba que a classe cachorro pode ter vários atributos:

Nome, Idade, Tamanho, Raça,  
Cor do pelo, Quantidade de  
dentes, Tamanho da cauda,  
Comida favorita.





# Abstração

A **Abstração** é a habilidade e a **capacidade** de se **modelar** conceitos, entidades, elementos, problemas e **características do mundo real**, de um domínio do problema em questão, levando-se em conta apenas os **detalhes** importantes para a **resolução do problema** e desprezando coisas que não têm importância no contexto.



**Vamos  
Praticar!!**

# Exercícios

## Programação Orientada a Objeto





# Exercícios

## Atividade 2:

Faça um algoritmo que crie uma classe candidatura que contenha o nome do candidato, a vaga desejada e a pretensão salarial. Imagine que exista duas possíveis vagas, TECNOLOGIA e GERAL. Consequentemente dois testes diferentes. Crie um método que identifique a vaga na qual o candidato se manifestou. Para cada objeto criado, faça uma breve apresentação do candidato e imprima qual teste deverá ser realizado (Teste de Tecnologia ou Teste Geral).



# Exercícios

```
package aula13;
public class Atividade2 {
    public static void main(String[] args) {
        Candidato c1 = new Candidato();
        c1.pessoaCandidata = "Ana Beatriz";
        c1.vaga = "TECNOLOGIA";
        c1.pretensaoSalarial = 12.000;
        c1.status();
        c1.enviarTesteTecnico();
        System.out.println(" ");

        Candidato c2 = new Candidato();
        c2.pessoaCandidata = "Caio Jose";
        c2.vaga = "GERAL";
        c2.pretensaoSalarial = 8.000;
        c2.status();
        c2.enviarTesteTecnico();
        System.out.println(" ");
    }
}
```



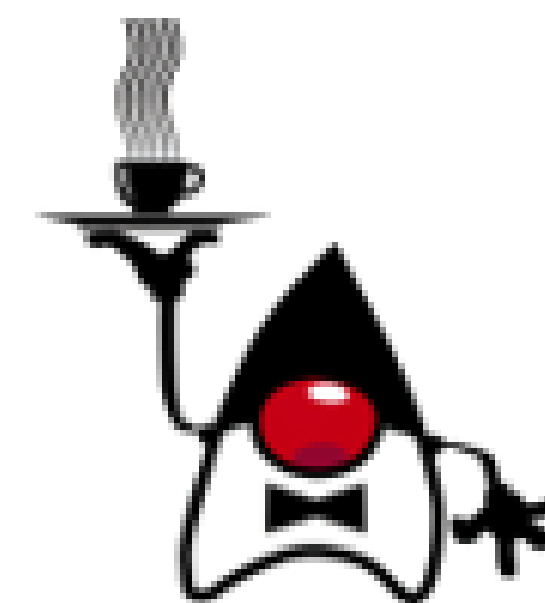
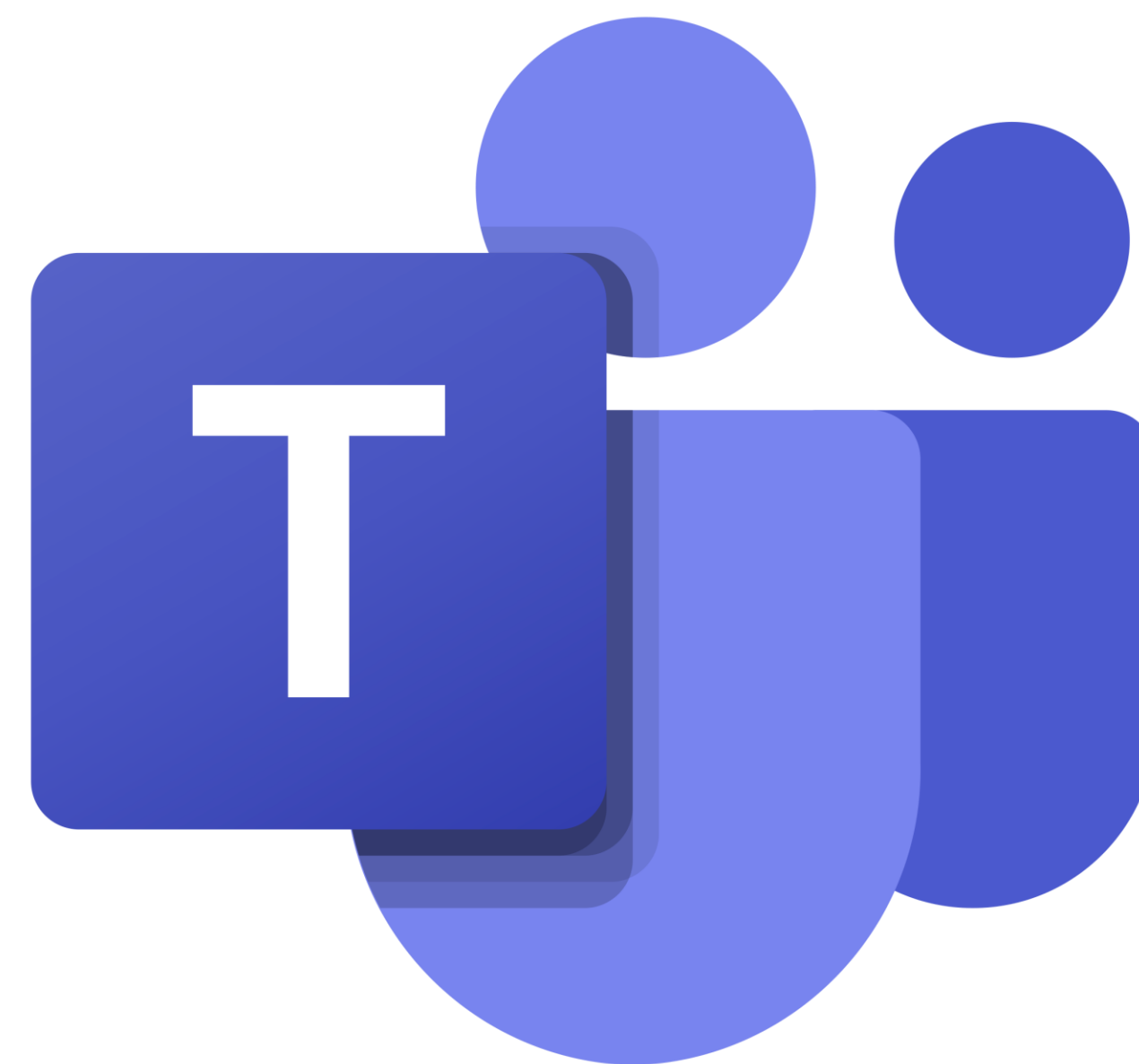


# Review e Preview





# Comunidade VNT



# Dica de hoje

O link abaixo apresenta uma breve explicação sobre o conceito de métodos e suas variadas formas de ser aplicada. Este é um tópico muito importante abordado em programação orientada a objeto.

<https://www.javatpoint.com/pt/m%C3%A9todo-em-java>

Boa leitura!!



# Referências

- [1] A. Goldman, F. Kon, Paulo J. S. Silva; Introdução à Ciência da Computação com Java e Orientação a Objetos (USP). 2006. Ed. USP.
- [2] Algoritmo e lógica de programação. Acessado julho/2022: <https://visualg3.com.br/>
- [3] G. Silveira; Algoritmos em Java; Ed. Casa do Código.
- [4] M. T. Goodrich, R. Tamassia; Estrutura de dados e algoritmos em Java. Ed Bookman. 2007.
- [5] Algoritmo e lógica de programação. Acessado julho/2022: <https://www.cursoemvideo.com/>
- [6] P. Silveira, R. Turini; Java 8 Prático: lambdas, streams e os novos recursos da linguagem. Ed. Casa do Código.
- [7] Linguagem Java: Curso acessado em agosto/2022: <https://www.udemy.com/>
- [8] Linguagem Java: Curso acessado em setembro/2022: <https://www.cursoemvideo.com/>

