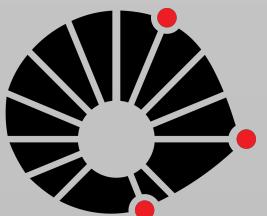


LINGUAGEM DE PROGRAMAÇÃO MULTIPLATAFORMA (LINGUAGEM JAVA)

Prof^a. Tania Basso

Colégio Técnico de Limeira



Orientação a Objeto

O que é um **objeto**?

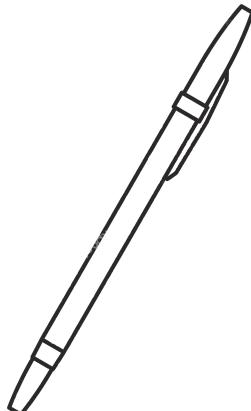
Definição → coisa material ou abstrata,
que pode ser percebida pelos sentidos e descrita por meio de
suas características, comportamentos e estado atual.



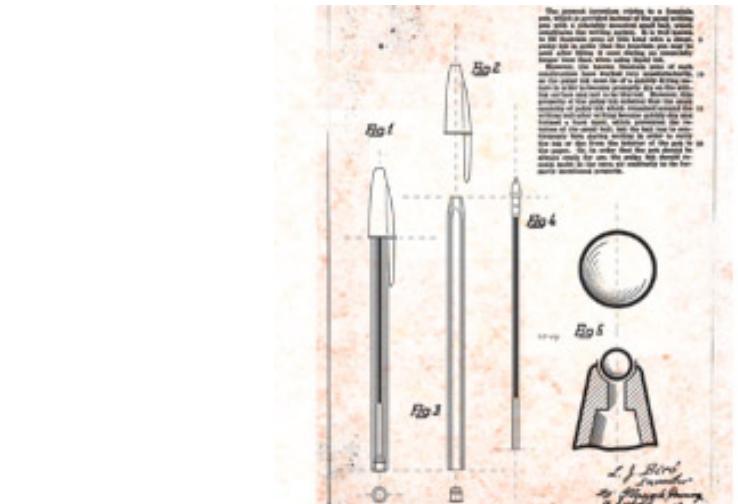
O que é uma Classe?

Definição → descrição de um molde (esqueleto) que especifica as propriedades (características) e comportamentos de objetos similares

Classe (molde para classificar canetas)



Caneta (objeto)



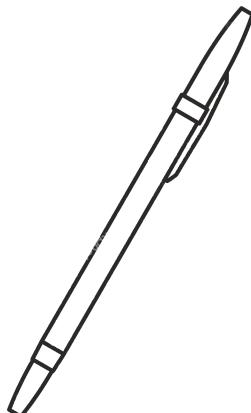
O jornalista húngaro László Biro criou uma caneta esferográfica após ver uma bola rolando sobre uma poça de água, deixando um rastro atrás dela. Desenvolveu um sistema em que uma esfera rolava por uma cavidade enquanto "pegava" a tinta do cartucho dentro da caneta e a transferia para o papel. Devido a entupimentos, vazamentos e má distribuição da tinta, nenhuma caneta da época fez sucesso como a Bic.

Fonte: <http://www.meionorte.com/blogs/robertofreitas/designs-inesqueciveis-113106>

Uma classe define **atributos** e **métodos**

Atributos → propriedades nomeadas de um objeto que armazenam o estado abstrato de cada objeto

Classe (molde para classificar canetas)



Caneta (objeto)



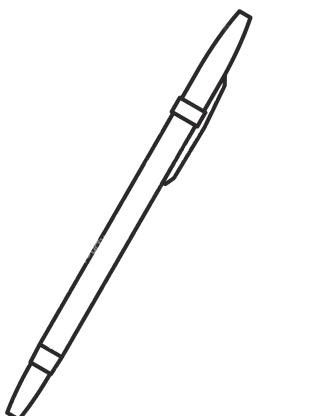
Atributos de uma caneta:

- Cor
- Modelo
- Ponta
- Tamanho
- Carga

Uma classe define **atributos** e **métodos**

Métodos → caracterizam o comportamento de um objeto. É o meio para acessar, modificar e manipular os atributos de um objeto

Classe (molde para classificar canetas)



Caneta (objeto)



Métodos de uma caneta:

- Escrever
- Rabiscar
- Tampar
- Destampar

Orientação a Objeto

Classe Caneta

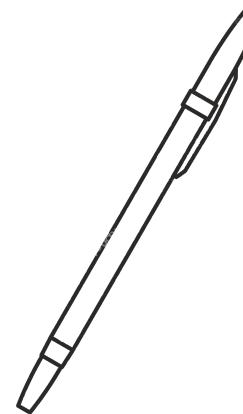
atributos

cor: caractere

modelo: caractere

ponta: real

tampada: lógico



métodos

escrever()

 se (**tampada**) então escreva ("erro")

 senão escreva ("teste")

fimMetodo

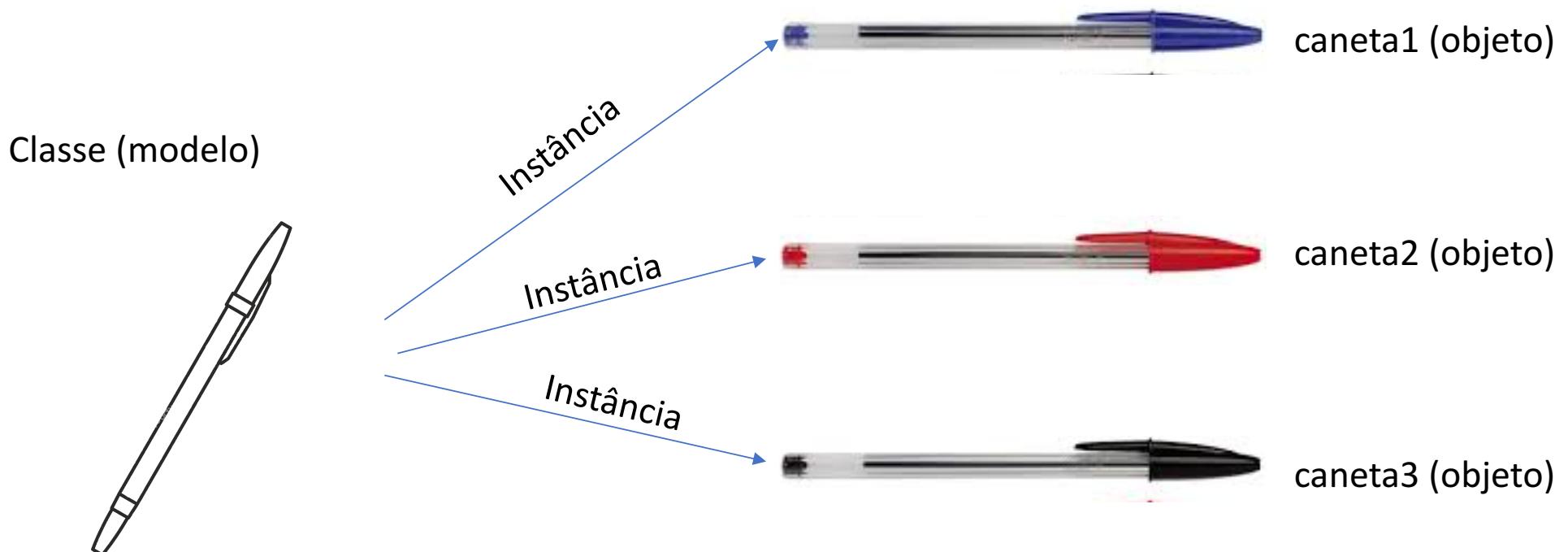
tampar()

tampada = verdadeiro

FimMetodo

FimClasse

Um **objeto** é uma **instância** de uma **classe**



Orientação a Objeto

Classe Caneta

atributos

cor: caractere

modelo: caractere

ponta: real

tampada: lógico

métodos

escrever()

```
    se (tampada) então escreva ("erro")
    senão escreva ("teste");
```

fimMetodo

tampar()

```
    tampada = verdadeiro
```

FimMetodo

FimClasse

Instanciando a classe Caneta:

caneta1 = nova Caneta

```
caneta1.cor = "azul";
caneta1.modelo = "bic";
caneta1.ponta = 0.5;
caneta1.tampada = true;
caneta1.escrever();
```



Mesmos
atributos,
valores
diferentes

caneta2 = nova Caneta

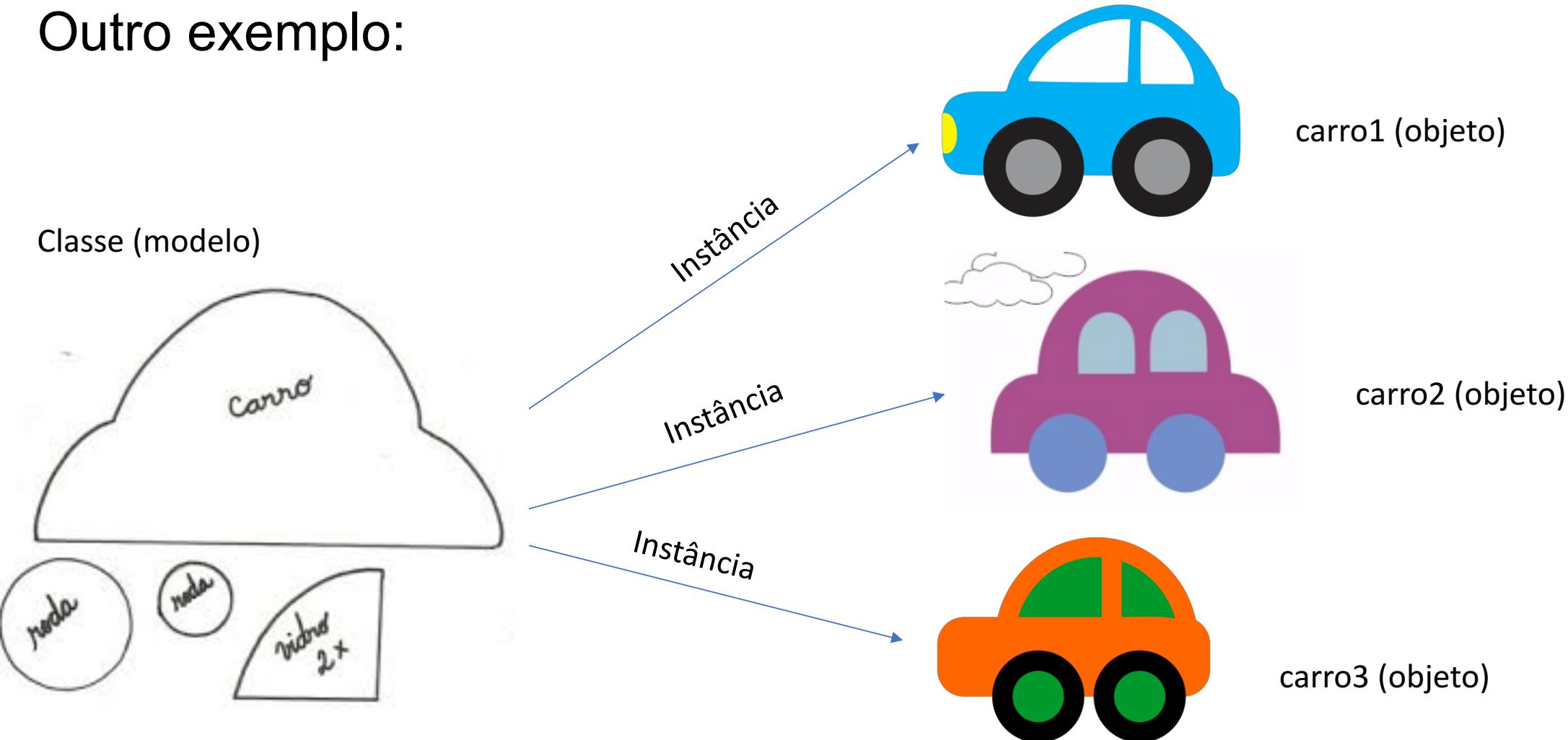
```
caneta2.cor = "vermelha";
caneta2.modelo = "pilot";
caneta2.ponta = 0.7;
caneta2.tampada = false;
caneta2.tampar();
```



Cada objeto
tem seu
estado!!

Orientação a Objeto

Outro exemplo:



Orientação a Objeto

Classe Carro

atributos

cor: caractere

placa: caractere

motor: real

modelo: caractere

métodos

andar()

 escreva ("andando")

fimMetodo

parar()

 escreva ("parando")

FimMetodo

FimClasse

Instanciando a classe Carro:

carro1 = novo Carro



carro1.cor = "azul";

carro1.placa = "ABC1234";

carro1.motor = 1.0;

carro1.modelo = gol;

carro1.andar();

carro1.parar();

carro2 = novo Carro



carro2.cor = "rosa";

carro2.placa = "XYZ3456";

carro2.motor = 1.6;

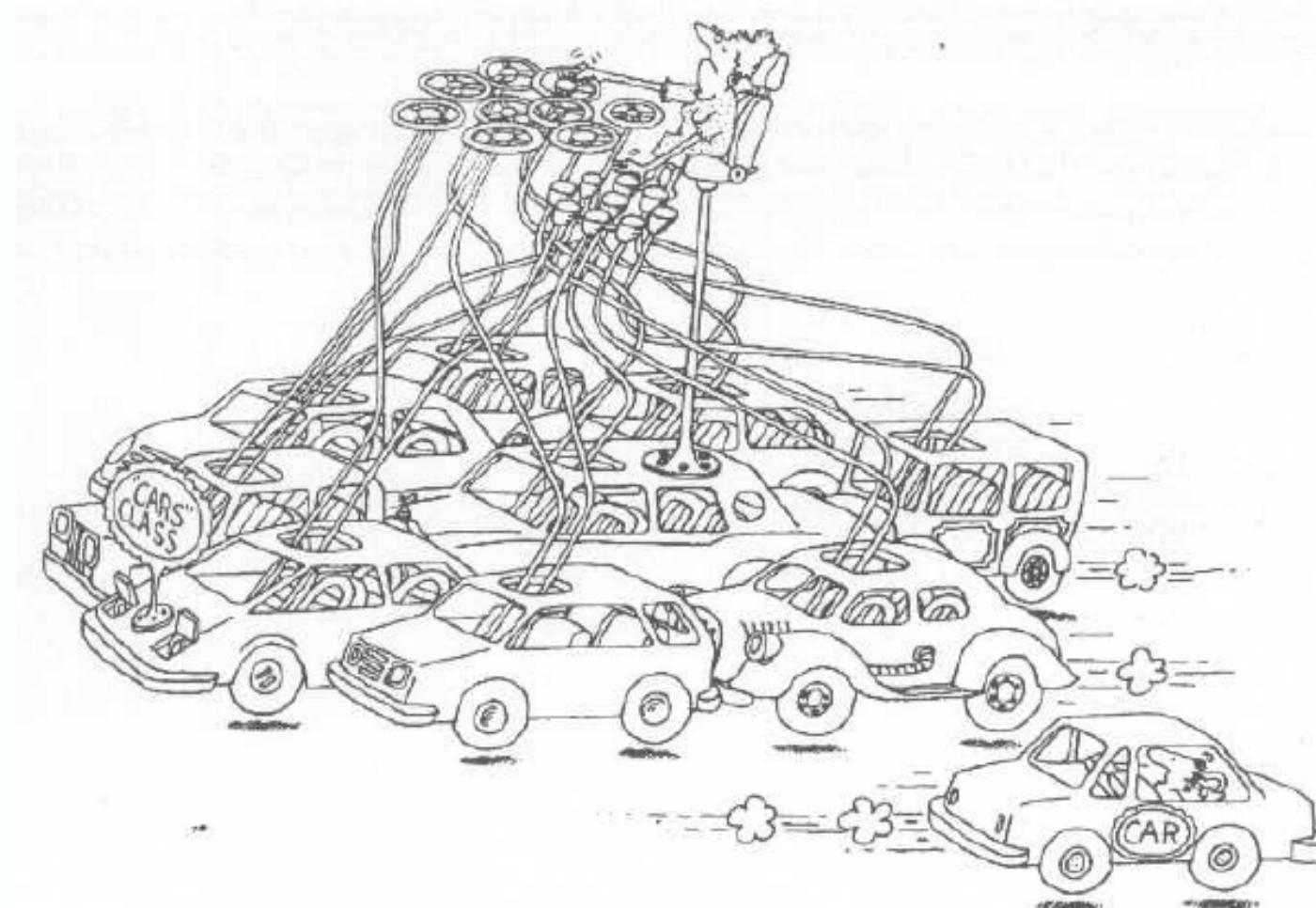
carro2.modelo = corsa;

carro2.andar();

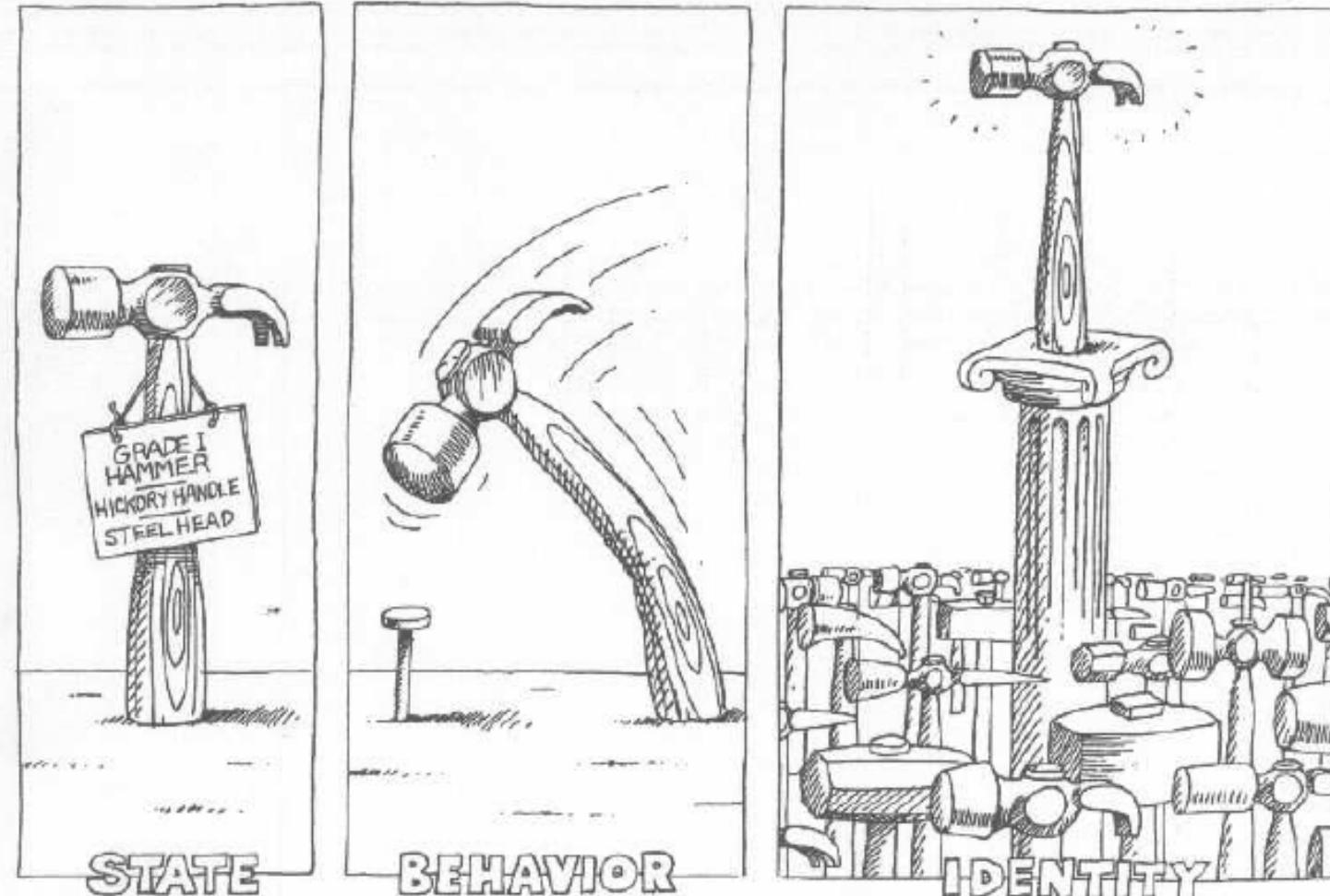
Mesmos
atributos,
valores
diferentes

Cada objeto
tem seu
estado!!

Classes (II)



Objetos (II)



Como fica, em Java, a definição de uma classe e instanciação de objetos?

```

package exemplocaneta;

/**
 *
 * @author taniabasso
 */
public class Caneta {
    String cor;
    String modelo;
    float ponta;
    boolean tampada;

    void escrever () {
        if (tampada)
            System.out.println("Erro - a caneta está tampada");
        else
            System.out.println("Escrevendo");
    }

    void destampar()
    {
        this.tampada = false;
    }

    void tampar()
    {
        this.tampada=true;
    }
}
  
```

```

public class ExemploCaneta {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Caneta caneta1 = new Caneta(); //instancia1
        caneta1.cor = "azul";
        caneta1.modelo = "bic";
        caneta1.ponta = 0.5f;
        caneta1.tampada = true;
        caneta1.escrever();
        caneta1.destampar();
        caneta1.escrever();

        Caneta caneta2 = new Caneta(); //instancia2
        caneta2.cor = "vermelha";
        caneta2.tampada = false;
        caneta2.escrever();
    }
}
  
```

Atenção às letras Maiúsculas e minúsculas!!

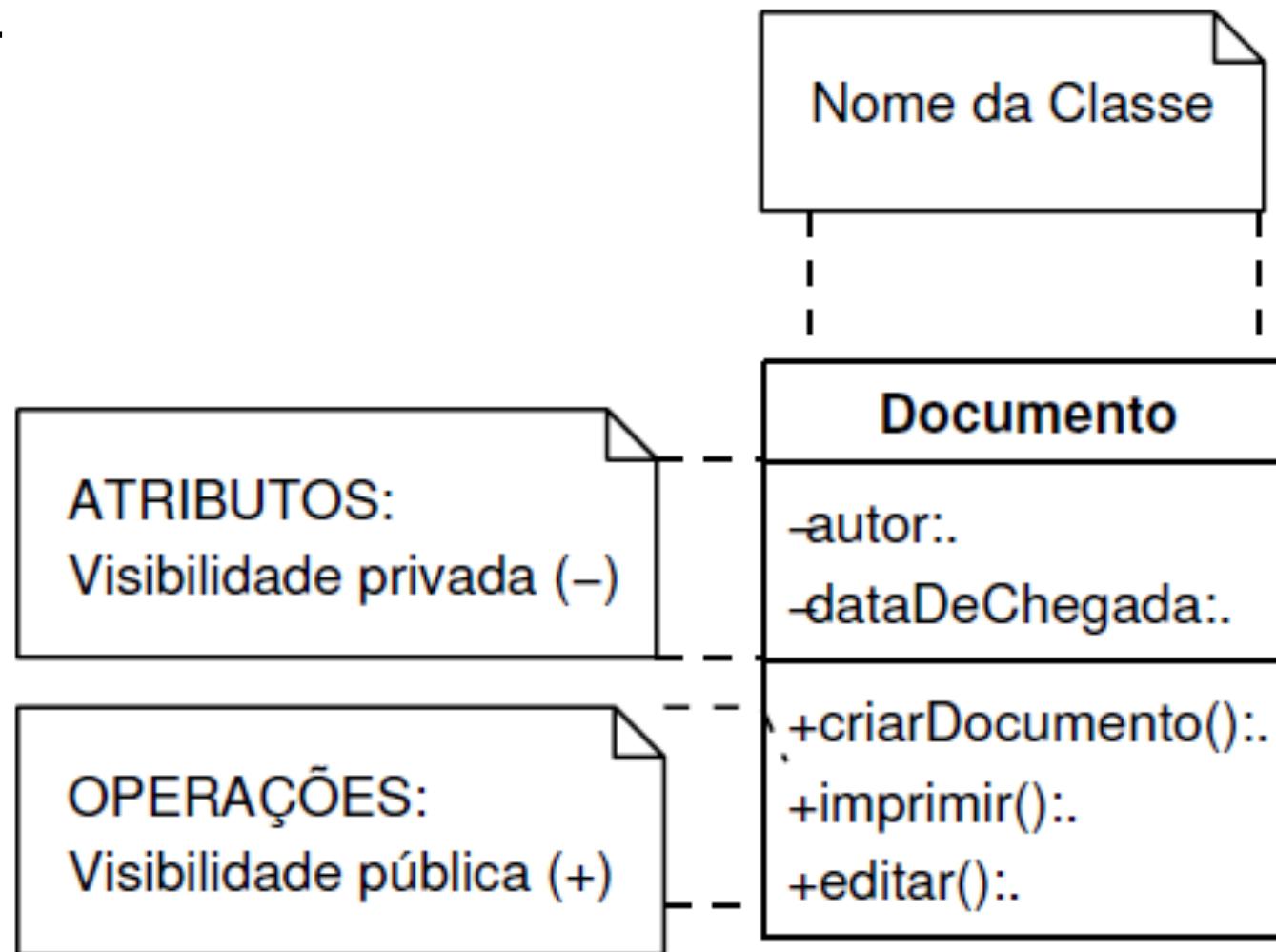
Exercícios

Defina, em Java, uma classe Aluno e crie diferentes instâncias.

No mesmo projeto, defina uma classe Disciplina e crie diferentes instâncias.

Defina, em Java, uma classe que represente algum objeto à sua escolha e crie instâncias dessa classe.

Notação UML



Objetos são entidades que **encapsulam** informação de estados ou dados, e um conjunto de operações associadas que manipulam esses dados.

O estado de um objeto é completamente **escondido** e protegido de outros objetos, a única maneira de examiná-lo é por meio de **métodos** definidos na **interface pública** do objeto

Objetos têm identidade única. Identidade é a propriedade de um objeto que o distingue de outros objetos.

Especificadores de acesso

Público → indica que atributos e métodos estão “disponíveis ao público”, podem ser chamados por outros métodos no programa e por métodos de outras classes



Privado → indica que atributos ou métodos são acessíveis somente a métodos da classe para a qual são declarados



Encapsulamento → atributos devem ser privados.

Acesso via **métodos** definidos na **interface pública** do objeto



```

public class Caneta {
    private String cor;
    private String modelo;
    private float ponta;
    private boolean tampada;

    public void escrever () {
        if (this.tampada)
            System.out.println("Erro – a caneta está tampada");
        else
            System.out.println("Escrevendo");
    }

    public void destampar()
    {
        this.tampada = false;
    }

    public void tampar()
    {
        this.tampada=true;
    }
}
  
```

```

public class ExemploCaneta {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Caneta caneta1 = new Caneta(); //instancia1
        caneta1.cor = "azul";
        caneta1.modelo = "bic";
        caneta1.ponta = 0.5f;
        caneta1.tampada = true;
        caneta1.escrever();
        caneta1.destampar();
        caneta1.escrever();

        Caneta caneta2 = new Caneta(); //instancia2
        caneta2.cor = "vermelha";
        caneta2.tampada = false;
        caneta2.escrever();
    }
}
  
```

ERRO!!

ERRO!!

Métodos GET / SET → interface pública

Atributos privados podem ser manipulados por meio de **métodos públicos get e set**

Set → recebe parâmetro e o atribui ao atributo. Normalmente não possui retorno (void) e pode fazer validação de dados antes da atribuição

Get → retorna atributo. Normalmente não possui argumento.

```
public class Caneta {  
  
    private String cor;  
    private String modelo;  
    private float ponta;  
    private boolean tampada;  
  
    public String getCor() {  
        return cor;  
    }  
  
    public void setCor(String cor) {  
        this.cor = cor;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

```
public class ExemploCaneta {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        Caneta caneta1 = new Caneta(); //instancia1  
        caneta1.setCor ("azul");  
        caneta1.setModelo ("bic");  
        caneta1.setPonta (0.5f);  
        caneta1.setTampada (true);  
        caneta1.escrever();  
        caneta1.destampar();  
        caneta1.escrever();  
        String corCaneta1 = caneta1.getCor();  
        System.out.println("A cor da caneta 1 é" + corCaneta1);  
  
        Caneta caneta2 = new Caneta(); //instancia2  
        caneta2.setCor ("vermelha");  
        caneta2.setTampada (false);  
        caneta2.setPonta (0.7f);  
        caneta2.escrever();  
        float pontaCaneta2 = caneta2.getPonta();  
        System.out.println("A ponta da caneta 2 é " + pontaCaneta2);  
    }  
}
```

Definição de métodos GET / SET no Netbeans

selecionar os atributos → botão direito → refatorar → encapsular campos

Lista de Campos a Serem Encapsulados:				
Campo	Criar Getter		Criar Setter	
cor : String	<input checked="" type="checkbox"/>	getCor() : String	<input checked="" type="checkbox"/>	setCor(String cor) : ...
modelo : String	<input checked="" type="checkbox"/>	getModelo() : String	<input checked="" type="checkbox"/>	setModelo(String m...)
ponta : float	<input checked="" type="checkbox"/>	getPonta() : float	<input checked="" type="checkbox"/>	setPonta(float pont...)
tampada : boolean	<input checked="" type="checkbox"/>	isTampada() : bool...	<input checked="" type="checkbox"/>	setTampada(boolean...)

O que são Construtores?

Métodos especiais chamados pelo sistema no momento da criação de um objeto.

Não retornam valores

Servem para **inicializar objetos** de forma organizada e consistente

Sempre tem o **mesmo nome da classe**

É chamado automaticamente no **momento da criação do objeto**

Construtores

Se nenhum construtor é definido para uma classe, o compilador cria um construtor default que não recebe argumentos

```
public static void main(String[] args) {  
    Caneta caneta1 = new Caneta(); //instância1
```

Construtor default (mesmo nome da classe)

Construtores → Podemos definir nossos construtores

```
public class Caneta {  
  
    private String cor;  
    private String modelo;  
    private float ponta;  
    private boolean tampada;  
  
    //construtor  
    public Caneta(String c, String m, float p, boolean t)  
    {  
        cor = c;  
        modelo = m;  
        ponta = p;  
        tampada = t;  
    }  
}
```

Construtor com argumentos



Inicializa atributos



Construtores → Podemos definir nossos construtores

```
/*
public static void main(String[] args) {
    Caneta caneta1 = new Caneta("azul", "bic", 0.5f, true); //instância1
    caneta1.escrever();
    caneta1.destampar();
    caneta1.escrever();
    String corCaneta1 = caneta1.getCor();
    System.out.println("A cor da caneta 1 é " + corCaneta1);
```

Instância chama construtor
com argumentos

Construtores → Podemos definir nossos construtores

```
/*
public static void main(String[] args) {
    Caneta caneta1 = new Caneta("azul", "bic", 0.5f, true); //instância1
    caneta1.escrever();
    caneta1.destampar();
    caneta1.escrever();
    String corCaneta1 = caneta1.getCor();
    System.out.println("A cor da caneta 1 é " + corCaneta1);
```

se passar um valor
negativo, por exemplo?

```
//construtor
public Caneta(String c, String m, float p, boolean t)
{
    cor = c;
    modelo = m;
    if (p<0)
        ponta = 0.4f;
    else
        ponta = p;
    tampada = t;
}
```

Atenção! Quando criamos outros construtores (sobrecarga), o construtor default (sem argumentos) não deve mais ser chamado
→ erro de sintaxe.

```
public static void main(String[] args) {  
    Caneta caneta1 = new Caneta("azul", "bic", 0.5f, true); //instância1  
    caneta1.escrever();  
    caneta1.destampar();  
    caneta1.escrever();  
    String corCaneta1 = caneta1.getCor();  
    System.out.println("A cor da caneta 1 é " + corCaneta1);  
  
    Caneta caneta2 = new Caneta();//instância2
```

Orientação a Objeto

Sobrecarga → Podemos definir vários construtores, mas com tipos e quantidades de argumentos diferentes

```
//construtor
public Caneta(String c, String m, float p, boolean t)
{
    cor = c;
    modelo = m;
    if (p<0)
        ponta = 0.4f;
    else
        ponta = p;
    tampada = t;
}

//construtor
public Caneta(String m)
{
    cor = "";
    modelo = m;
    ponta = 0;
    tampada = false;
}

//construtor
public Caneta(String m, boolean t)
{
    cor = "";
    modelo = m;
    ponta = 0;
    tampada = t;
}
```

Sobrecarga → também vale para métodos da classe

```
public void escrever ()  
{  
    if (this.isTampada())  
        System.out.println("Erro – a caneta está tampada");  
    else  
        System.out.println("Escrevendo");  
  
public void escrever (String cor)  
{  
    System.out.println("Escrevendo com a caneta "+ cor);  
  
public void escrever (String cor, float ponta)  
{  
    System.out.println("Escrevendo com a caneta "+ cor + " ponta " + ponta);  
}
```

Método tem mesmo nome,
mas quantidade e tipos de
argumentos diferentes

Sobrecarga → também vale para métodos da classe

```
public static void main(String[] args) {  
    Caneta caneta1 = new Caneta("azul", "bic", 0.5f, true); //instância1  
    caneta1.escrever();  
    caneta1.escrever("verde");  
    caneta1.escrever("rosa", 0.7f);
```

Chamada aos métodos

run:

Erro – a caneta está tampada
Escrevendo com a caneta verde
Escrevendo com a caneta rosa ponta 0.7

Exercício

Crie uma classe **Retângulo**. A classe deve ter os atributos **base** e **altura**; cada um deles é configurado com valor *default* 1. Ela deve ter métodos **get** e **set** para os atributos e métodos que calculam e retornam o **perímetro** e a **área** do retângulo. Os métodos **set** devem verificar se tanto base quanto altura têm valores entre 1 e 20. Escreva um programa para testar a classe **Retângulo**.

Dúvidas?



tbasso@unicamp.br