

Análise e Projeto de Desenvolvimento de Sistemas

APDS

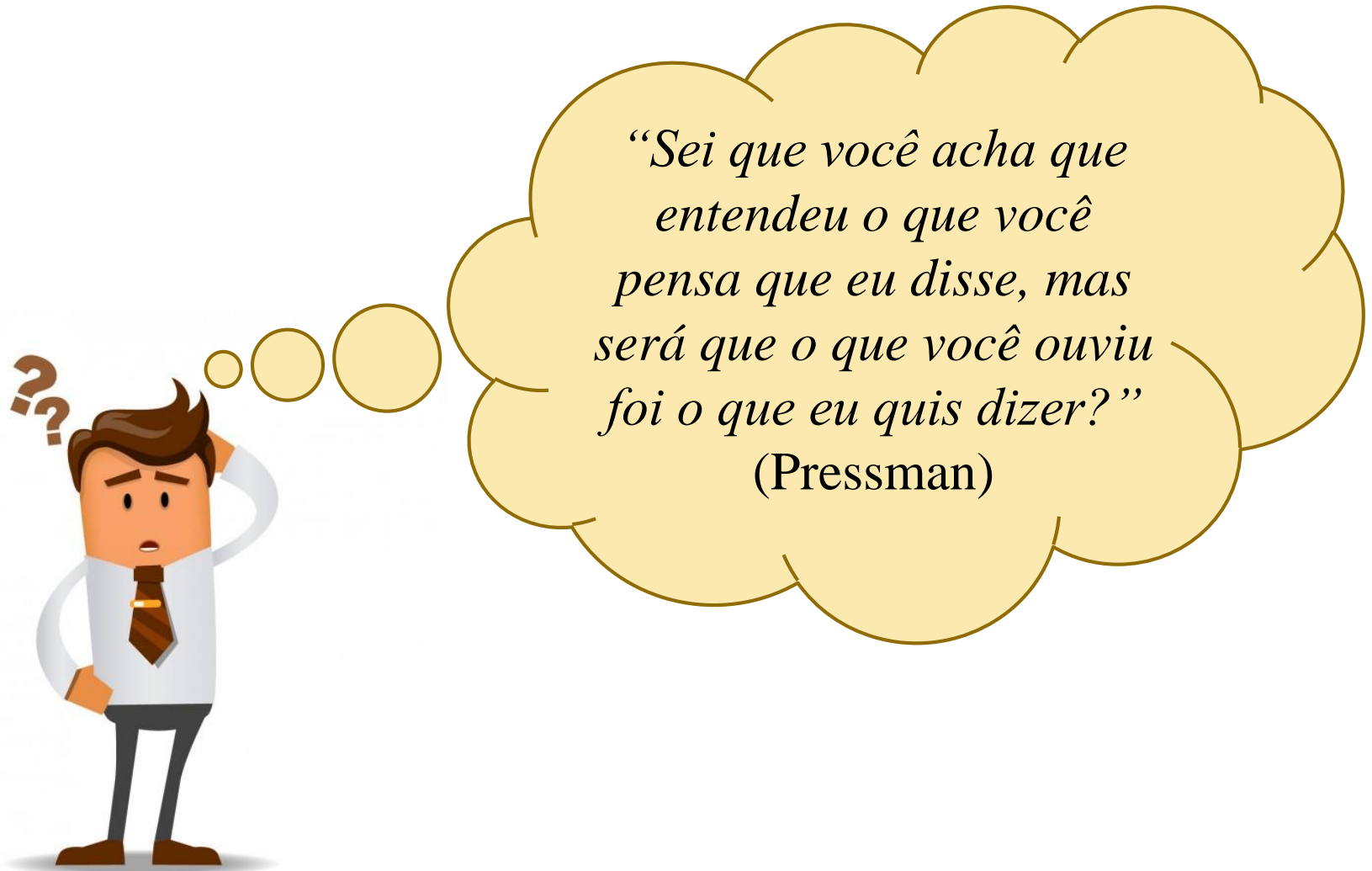
Aula 1

Universidade Federal do Rio Grande do Norte
Unidade Acadêmica Especializada em Ciências Agrárias
Escola Agrícola de Jundiá
Técnico de Informática
Profa. Alessandra Mendes



E agora?

O Problema



O que se quer evitar...



Como o cliente explicou...



Como o líder de projeto entendeu...



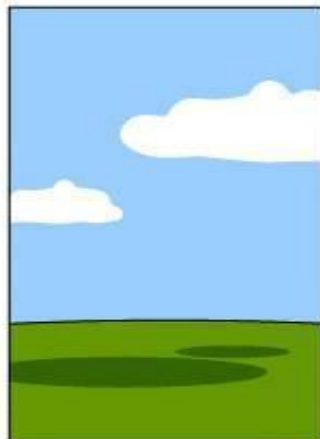
Como o analista projetou...



Como o programador construiu...



Como o Consultor de Negócios descreveu...



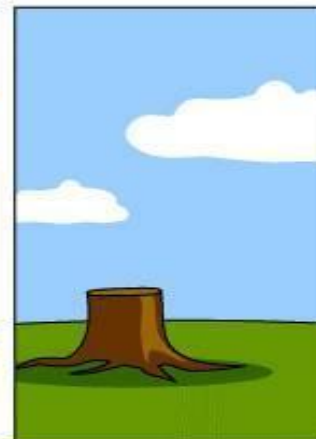
Como o projeto foi documentado...



Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...

Problema x Abstração

“Para resolver um problema é necessário escolher uma abstração da realidade” (Almeida, 2010)

- ▶ A abstração é o “processo mental que consiste em escolher ou isolar um **aspecto** determinado de um estado de coisas relativamente complexo, a fim de **simplificar** a sua avaliação, classificação ou para permitir a comunicação do mesmo” (Houaiss, 2006)
- ▶ Abstrações ajudam a gerenciar a complexidade do software (Shaw, 1984)

Tipo abstrato de dados

- ▶ “O termo 'tipo abstrato de dados' se refere ao **conceito** matemático básico (não considera aspectos de implementação) que define um **tipo** de dados” (Tenenbaum, 1990)
- ▶ “Um tipo abstrato de dados define uma **classe** de objetos abstratos que é completamente **caracterizada** pelas **operações** disponíveis nestes objetos. Isto significa que um tipo abstrato de dados pode ser definido pela definição e caracterização das operações daquele tipo.” (Liskov, 1974)

Sistemas

- ▶ Sistemas são **complexos**
 - ▶ Ciclo de vida longo
 - ▶ Muito difícil um único indivíduo entender todo o sistema.
- ▶ Podemos **gerenciar** a complexidade, não podemos eliminá-la.
- ▶ O **domínio do problema** é complexo
 - ▶ Necessário pensar na evolução do sistema

COMO FUNCIONA A CABEÇA DE UM PROGRAMADOR:



Desenvolvimento de Sistemas

- ▶ **Análise:** define **o que** deve ser feito
 - ▶ Diagrama de Casos de Uso e Modelo Conceitual
- ▶ **Projeto:** define **como** deve ser feito
 - ▶ Diagrama de Classes
- ▶ **Implementação**
 - ▶ Escreve em uma linguagem de programação (**faz**)





Análise e Projeto

Por que Análise e Projetos?

- ▶ Conhecer uma linguagem de programação orientada a objetos (OO) não é suficiente...
 - ▶ Muitas vezes, programa-se de forma estruturada utilizando uma linguagem OO
- ▶ Deve-se **pensar** OO
 - ▶ Para isso, deve-se aprender Análise e Projeto OO...
 - ▶ ... ou seja, análise e projeto de software segundo o paradigma de orientação a objetos

O que é Análise?

- ▶ A análise consiste nas atividades necessárias para **entender** o domínio do problema.
- ▶ Trata-se de uma atividade de **investigação**.

O que deve ser feito?

- ▶ A análise consiste de atividades feitas **com** e **para** o cliente (análise de requisitos).
- ▶ A informação produzida na análise deve ser **discutida e aprovada pelo cliente**
 - ▶ Invade-se um pouco o espaço da solução...
 - ▶ Interface do usuário, por exemplo.

O que deve ser feito de acordo com o cliente?

O que é Projeto?

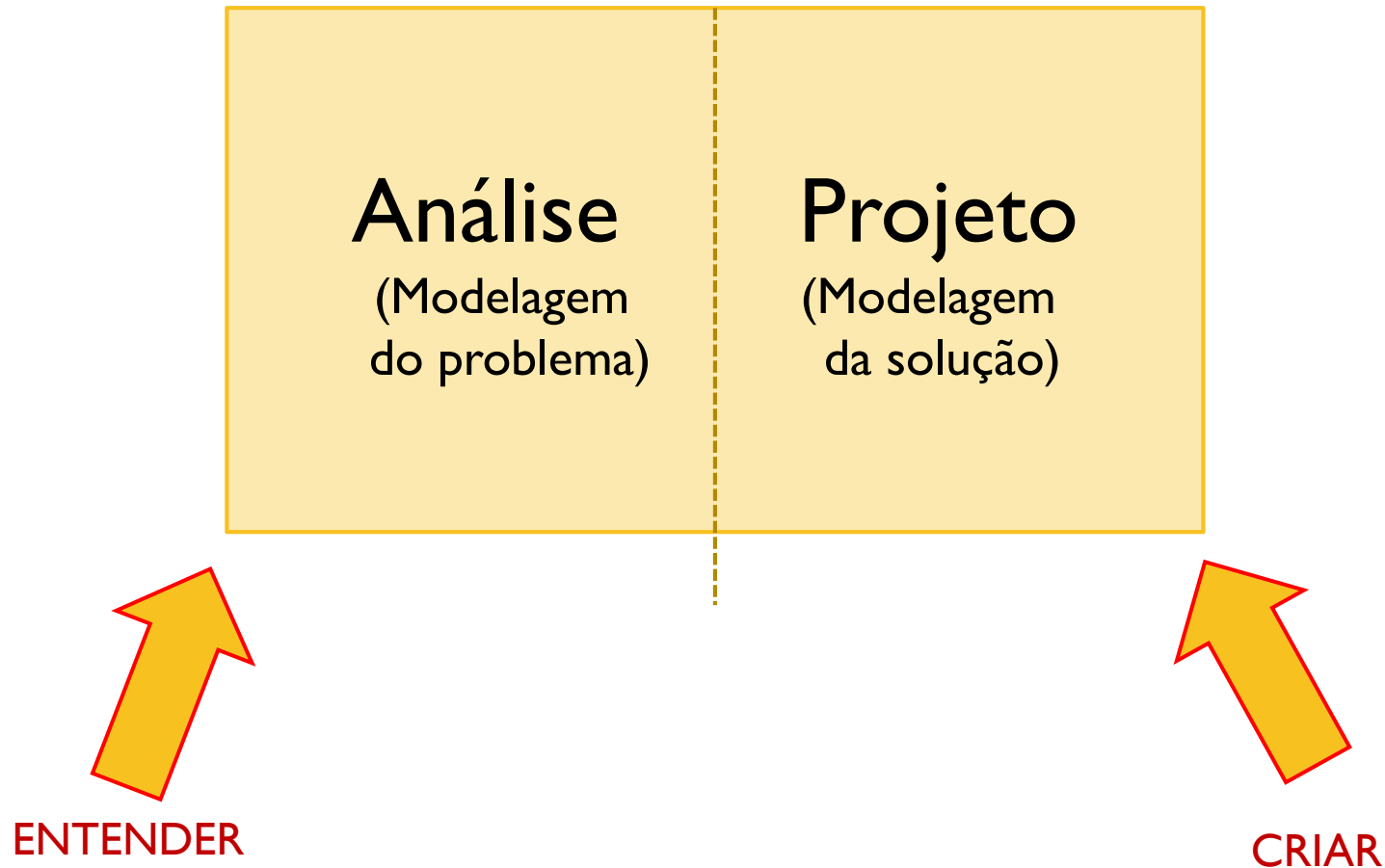
- ▶ O projeto modela a solução e consiste das atividades de criação.
- ▶ Trata-se de uma atividade de **resolução**.

Como pode ser feito?

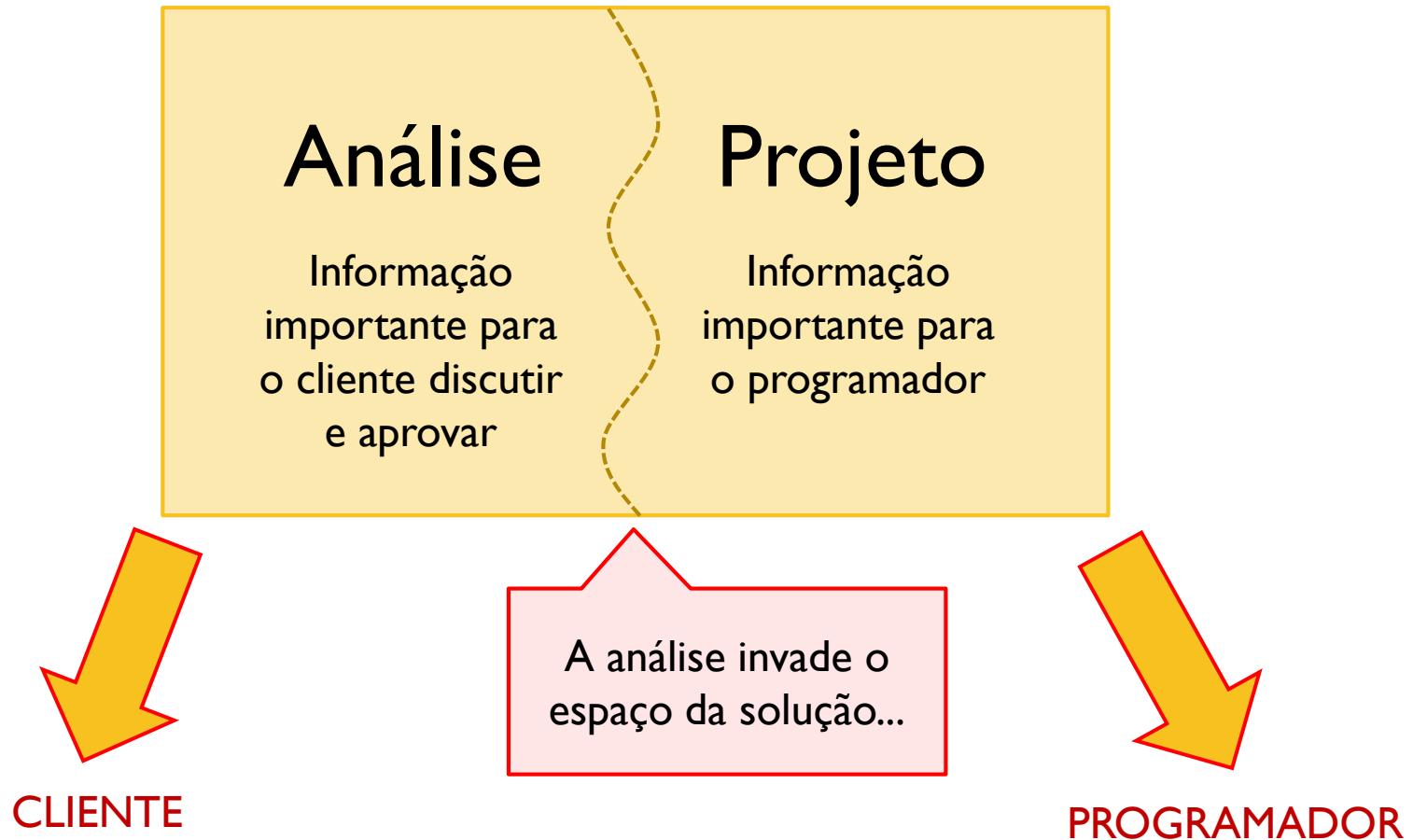
- ▶ O Projeto inclui as atividades que resultam em informação que interessa apenas ao programador.
- ▶ A atividade de projeto serve como base para a atividade de programação (construção).

Programador, veja como deve ser feito!

Análise e Projeto

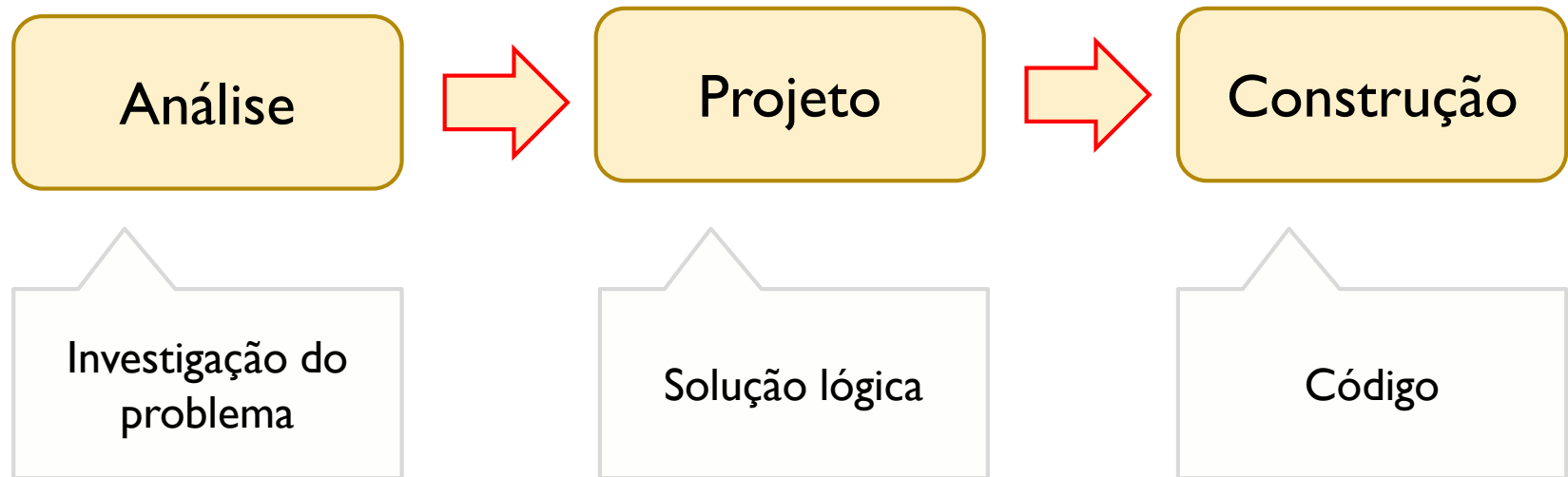


Análise e Projeto



Análise, Projeto e Construção

► Resumindo...



- **Análise e projetos criam modelos;**
- **Análise e Projetos Orientados a Objetos criam modelos OO.**



Orientação a Objetos

Por que Orientação a Objetos?

► Modelagem OO

- O ser humano conhece o mundo e gerencia sua complexidade através de **objetos**;
- A partir do conceito de objetos desenvolvemos nossa **cognição**, ou seja, adquirimos conhecimento.

► Exemplos de objetos

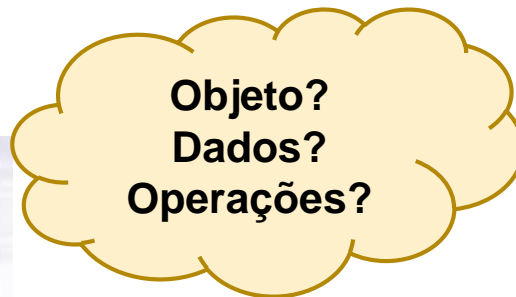
- Bola, carro, camisa, relógio, casa, gato, conta bancária, poema, cachorro, pessoa, etc.



O que são objetos?

- ▶ Um objeto representa um **item identificável**, uma unidade, ou entidade, individual, seja **real** ou **abstrato**, com uma regra bem definida;
- ▶ O que se apresenta à percepção com um caráter fixo e estável.

OBJETO = DADOS + OPERAÇÕES



Objetos = Dados + Operações

► Dados

- Os dados representam as **características**;
- São chamados de **atributos**;
- São as variáveis de instância.

► Operações

- As operações definem o **comportamento**;
- São os **métodos** de um objeto;
- São as funções que são executadas em um objeto.

Cachorro

Quais são suas características?

- Nome
- Raça
- Idade

Qual o seu comportamento?

- Come
- Dorme
- Passeia



Objetos possuem...

▶ Estado

- ▶ Representado pelos **valores dos atributos** de um objeto.

▶ Comportamento

- ▶ Definido pelo **conjunto de métodos** do objeto.
 - ▶ O estado representa o resultado cumulativo do seu comportamento.

▶ Identidade

- ▶ Forma pela qual conhecemos o objeto, é a **referência** ao objeto.

Meu Cachorro

Atributos:

- Nome: **Magnífico**
- Raça: **Beagle**
- Idade: **2 anos**

Métodos:

- **Come**
- **Dorme**
- **Passeia**



Classes e Objetos

▶ Classe

- ▶ Tipo definido pelo usuário que possui especificações (características ou estados, comportamentos e identidade);
- ▶ Fornecem a estrutura para a **construção** de objetos, é onde **conceituamos** o objeto;
- ▶ **Define** os atributos e métodos.
- ▶ O objeto é uma **instância** de uma classe

| Cachorro |
|---|
| - nome : String - raça : String - idade : int |
| + comer() : void + dormir() : void + passear() : void |

powered by Astah

Dinossauro

Diego



Classes e Objetos

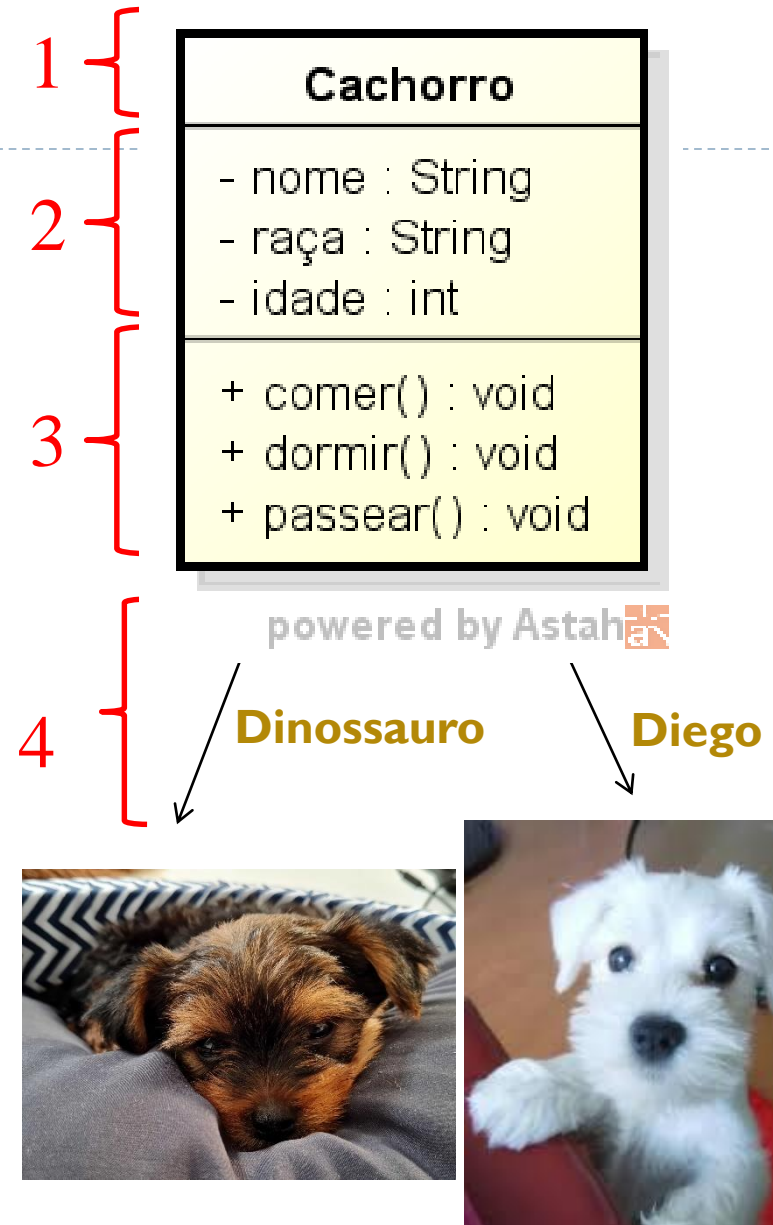
1. Nome da classe

2. Atributos

3. Métodos

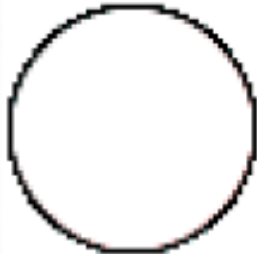
4. Objetos

- ▶ Duas instâncias diferentes da mesma classe



Classes e Objetos

► Exemplo: **Classe Esfera**

| Classe Esfera | |
|---|---------------------------------|
| Atributos (nome, tipo) | |
|  | (peso , real) |
| | (raio , real) |
| | (elasticidade , string) |
| | (cor , color) |
| Comportamento | |
| aumentar, diminuir, se mover | |


Classes e Objetos

► Exemplo: **Objeto Esfera**

| Objeto Esfera | |
|---|-------------------------------|
| Atributos (nome, valor) | |
|  | (peso , 200 g) |
| | (raio , 60 cm) |
| | (elasticidade , alta) |
| | (cor , vermelha) |
| Comportamento | |
| aumentar, diminuir, se mover | |


Classes e Objetos

► Exemplo: **Classe Financiamento**

| Classe Financiamento | |
|---|-------------------------------|
| Atributos (nome, tipo) | |
|  | (valor, real) |
| | (número de parcelas, inteiro) |
| | (percentual de juros, real) |
| | |
| Comportamento | |
| calcula parcela | |

Classes e Objetos

► Exemplo: **Objeto Financiamento**

| Objeto Financiamento | |
|---|---------------------------|
| Atributos (nome, valor) | |
|  | (valor, R\$ 150) |
| | (número de parcelas, 3) |
| | (percentual de juros, 1%) |
| | |
| Comportamento | |
| calcula parcela | |

Classes e Objetos

▶ Classe **Lâmpada**

▶ Atributos

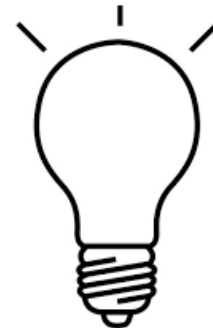
- ▶ potencia (int)
- ▶ ligada (boolean)

▶ Métodos

- ▶ ligar
- ▶ desligar
- ▶ estaLigada
- ▶ verificarPotencia
- ▶ alterarPotencia

| Lâmpada |
|--|
| - ligada : boolean - potencia : int |
| + ligar() : void + desligar() : void + estaLigada() : boolean + verificarPotencia() : int + alterarPotencia(novaPotencia : int) : void |

powered by Astah



Classes e Objetos

Classe Lâmpada

Em java

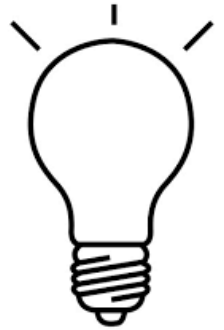
| Lâmpada |
|--|
| - ligada : boolean - potencia : int |
| + ligar() : void + desligar() : void + estaLigada() : boolean + verificarPotencia() : int + alterarPotencia(novaPotencia : int) : void |

powered by Astah

Atributos

Métodos

```
public class Lâmpada {  
    private boolean ligada;  
    private int potencia;  
  
    public void ligar(){  
        ligada = true;  
    }  
    public void desligar(){  
        ligada = false;  
    }  
    public boolean estaLigada(){  
        return ligada;  
    }  
    public int verificarPotencia(){  
        return potencia;  
    }  
    public void alterarPotencia(int novaPotencia){  
        potencia = novaPotencia;  
    }  
}
```



Classes e Objetos

Classe

| Lâmpada |
|--|
| - ligada : boolean - potencia : int |
| + ligar() : void + desligar() : void + estaLigada() : boolean + verificarPotencia() : int + alterarPotencia(novaPotencia : int) : void |

powered by Astah

Objetos



| Lâmpada 1 |
|---------------------------------|
| Ligada = true Potencia = 10v |



| Lâmpada 2 |
|---------------------------------|
| Ligada = true Potencia = 50v |



Vamos à prática...

Exercício 1: Lâmpada

- ▶ Declarar a classe Lâmpada;

| Lâmpada |
|--|
| - ligada : boolean - potencia : int |
| + ligar() : void + desligar() : void + estaLigada() : boolean + verificarPotencia() : int + alterarPotencia(novaPotencia : int) : void |

powered by Astah

- ▶ Declarar uma classe Principal, contendo:
 - ▶ Um objeto da classe Lâmpada
 - ▶ Um menu que apresente as opções referentes aos métodos da classe Lâmpada, peça a opção do usuário e implemente-a de acordo com o que ele escolheu.

Exercício 2: Elevador

- ▶ Crie a classe Elevador para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o andar atual (térreo = 0), total de andares no prédio (desconsiderando o térreo), capacidade do elevador e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:
 - ▶ Inicializa: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazios);
 - ▶ Entra: acrescenta uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
 - ▶ Sai: remove uma pessoa do elevador (só deve remover se houver alguém dentro dele);
 - ▶ Sobe: sobe um andar (não deve subir se já estiver no último andar);
 - ▶ Desce: desce um andar (não deve descer se já estiver no térreo);

Exercício 2: Elevador

- ▶ Declarar uma classe Principal, contendo:
 - ▶ Um objeto da classe Elevador
 - ▶ Um menu que apresente as opções referentes aos métodos da classe Elevador, peça a opção do usuário e implemente-a de acordo com o que ele escolheu.



Dúvidas?