Primeiros Passos com Orientação a Objetos

Thayse Onofrio

Thayse Onofrio

25 anos

Publicitária

Estudante de Sistemas para a Internet

Consultora de Desenvolvimento de Software na Thoughtworks

O que é Programação Orientada a Objetos?

O básico

A ideia é usar objetos para modelar as coisas do mundo real que queremos representar nos nossos programas.

Objetos contém **informações** sobre a coisa que você está representando e **funcionalidades** (ou comportamentos) que possui.

Programação estruturada

Processos

Dividida em funções

Menos reusabilidade

Menos flexibilidade

Programação orientada a objetos

Dados

Objetos

Mais reusabilidade e menos dependência

Mais flexibilidade

Definindo o modelo de um objeto

Classe: Pessoa

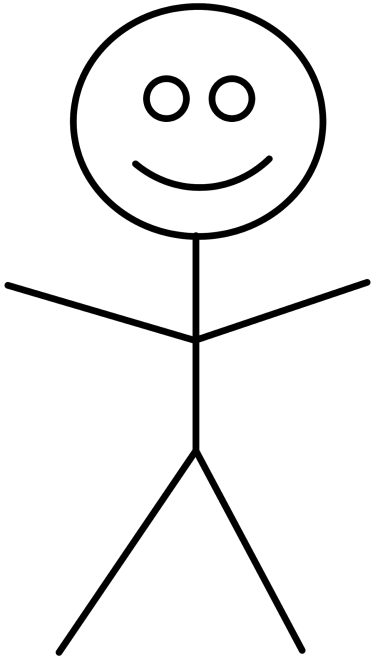
Nome

Idade

Profissão

Apresentar{"Oi! Meu nome é {nome} e tenho

{idade} anos"}



Abstração

Criar um modelo simples de algo complexo.

Classes

Uma classe define atributos e comportamentos.

É um modelo genérico dos objetos que queremos criar.

Criando objetos

Agora que temos um modelo (uma **classe**), podemos criar **instâncias de objetos**, que contém as informações e funcionalidades definidos na classe.

Criando pessoas

Objeto: Pessoa 1

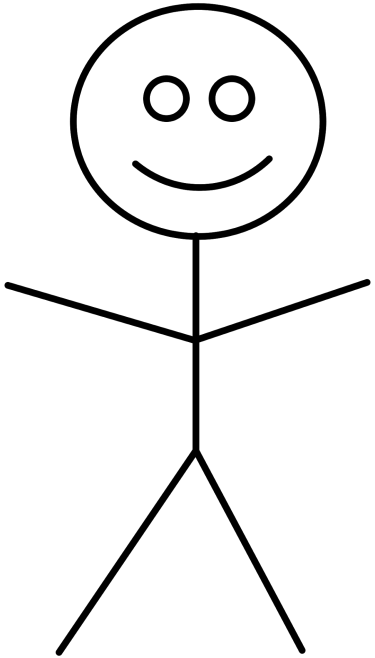
Nome: Beyonce

Idade: 37

Profissão: Rainha

Apresentar{"Oi! Meu nome é Beyonce e tenho 37

anos"}



Criando pessoas

Objeto: Pessoa 2

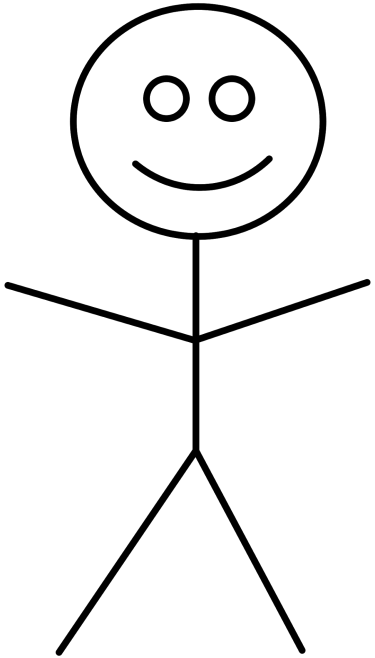
Nome: Lady Gaga

Idade: 32

Profissão: Artista

Apresentar{"Oi! Meu nome é Lady Gaga e tenho

32 anos"}



Instanciando

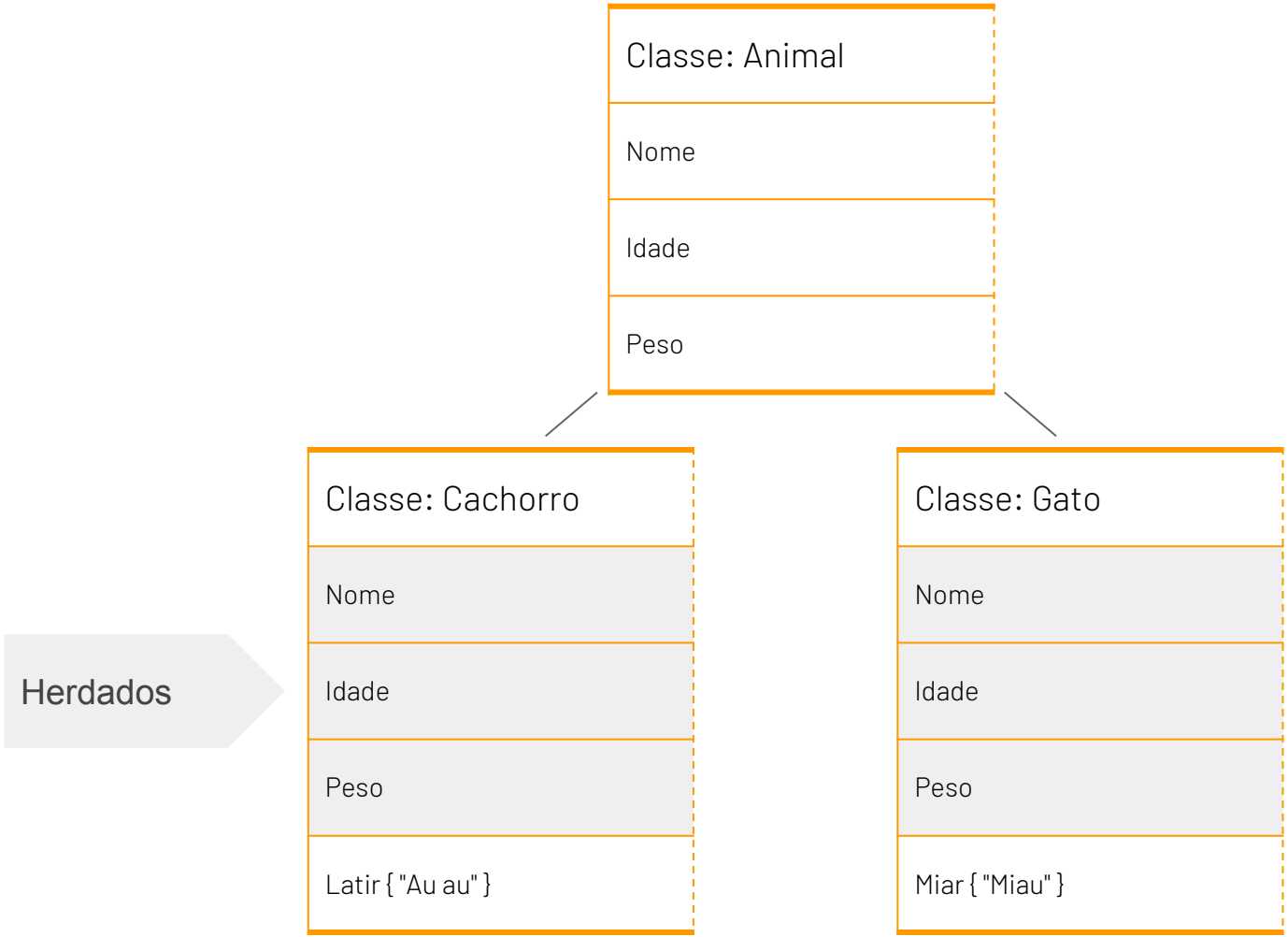
Quando uma instância de um objeto é criado a partir de uma classe, nós chamamos esse processo da Instanciar.

Nós **instanciamos** a classe.

Herança

Em Orientação a Objetos, podemos criar classes baseadas em outras classes. Essas classes filhas **herdam** as informações e funcionalidades da classe pai.

Podemos reutilizar o que for comum entre as classes, sem ter que duplicar o código. Se a classe filha tiver algo de diferente, podemos definir isso diretamente nela.


Classe: Animal
Nome
Idade
Peso
Classe: Cachorro
Nome
Idade
Peso
Latir { "Au au" }
Classe: Gato
Nome
Idade
Peso
Miar { "Miau" }
Herdados

Instâncias das classes filhas

Classe: Gato
Nome
Idade
Peso
Miar { "Miau" }

Objeto: Gato 1
Nome: Lily
Idade: 2
Peso: 3kg
Miar { "Miau" }

Fazendo tudo isso em Java

```java
public class Animal {
    public String nome;
    public int idade;
    public int peso;

    public Animal(String nome, int idade, int peso) {
        this.nome = nome;
        this.idade = idade;
        this.peso = peso;
    }
}
```

```java
public class Cachorro extends Animal {

    public Cachorro(String nome, int idade, int peso) {
        super(nome, idade, peso);
    }

    public void latir() {
        System.out.println("Au au");
    }
}
```

```java
public class Gato extends Animal {

    public Gato(String nome, int idade, int peso) {
        super(nome, idade, peso);
    }

    public void miar() {
        System.out.println("Miar");
    }
}
```

```
Cachorro cachorro1 = new Cachorro( nome: "Tobby",  idade: 4,  peso: 8);

cachorro1.latir();

Gato gato1 = new Gato( nome: "Lily",  idade: 2,  peso: 3);

gato1.miar();
```

Outros conceitos importantes

Encapsulamento

O encapsulamento é atingido quando cada objeto mantém seus dados **privados** dentro de uma classe. Outros objetos não tem acesso direto a esses dados.

Os outros objetos só podem acessar os **métodos públicos** da classe.

```java
public class Pessoa {

    private String nome;

    public Pessoa(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }
}
```

```
Pessoa pessoa = new Pessoa( nome: "Beyonce",  idade: 37);

pessoa.nome;
```

'nome' has private access in 'com.oop.Pessoa'

```
pessoa.getNome();
```

Polimorfismo

“Muitas formas”

Os objetos filhos herdam as características dos seus pais, mas às vezes é necessário que ações para um mesmo método sejam diferentes.

```java
public class Animal {
    public String nome;
    public int idade;
    public int peso;

    public Animal(String nome, int idade, int peso) {
        this.nome = nome;
        this.idade = idade;
        this.peso = peso;
    }

    public void fazBarulho() {
        System.out.println("Animal fazendo algum som");
    }
}
```

```java
public class Cachorro extends Animal {

    public Cachorro(String nome, int idade, int peso) {
        super(nome, idade, peso);
    }

    @Override
    public void fazBarulho() {
        System.out.println("Au au");
    }
}
```

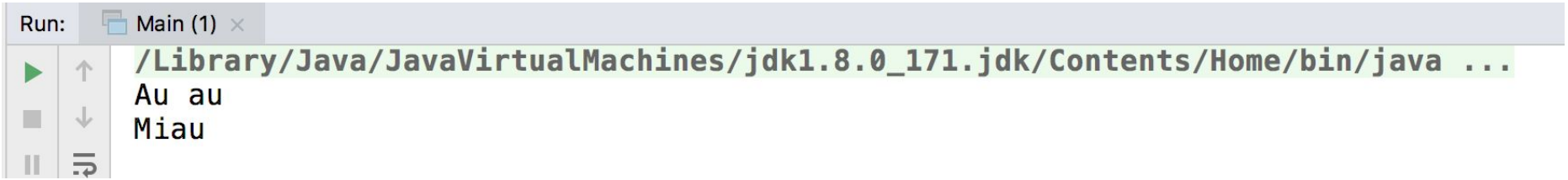
```java
public class Gato extends Animal {

    public Gato(String nome, int idade, int peso) {
        super(nome, idade, peso);
    }

    @Override
    public void fazBarulho() {
        System.out.println("Miau");
    }
}
```

```java
Cachorro cachorro1 = new Cachorro( nome: "Tobby",  idade: 4,  peso: 8);
//cachorro1.latir();
cachorro1.fazBarulho();

Gato gato1 = new Gato( nome: "Lily",  idade: 2,  peso: 3);
//gato1.miar();
gato1.fazBarulho();
```



4 Pilares da Orientação a Objetos

Abstração

Encapsulamento

Herança

Polimorfismo

Principais vantagens

Próximo a realidade

Reutilização de código

Sistemas complexos

Leitura e Manutenção do código

Onde continuar estudando
em Português e de graça

Conceitos básicos de Orientação a Objetos

Apostila Java e Orientação a Objetos

Curso Orientação a Objetos com Java

Java básico - Loiane Groner ❤

Obrigada!

dúvidas?

Thayse Onofrio
tonofrio@thoughtworks.com