Juntos, somos únicos.

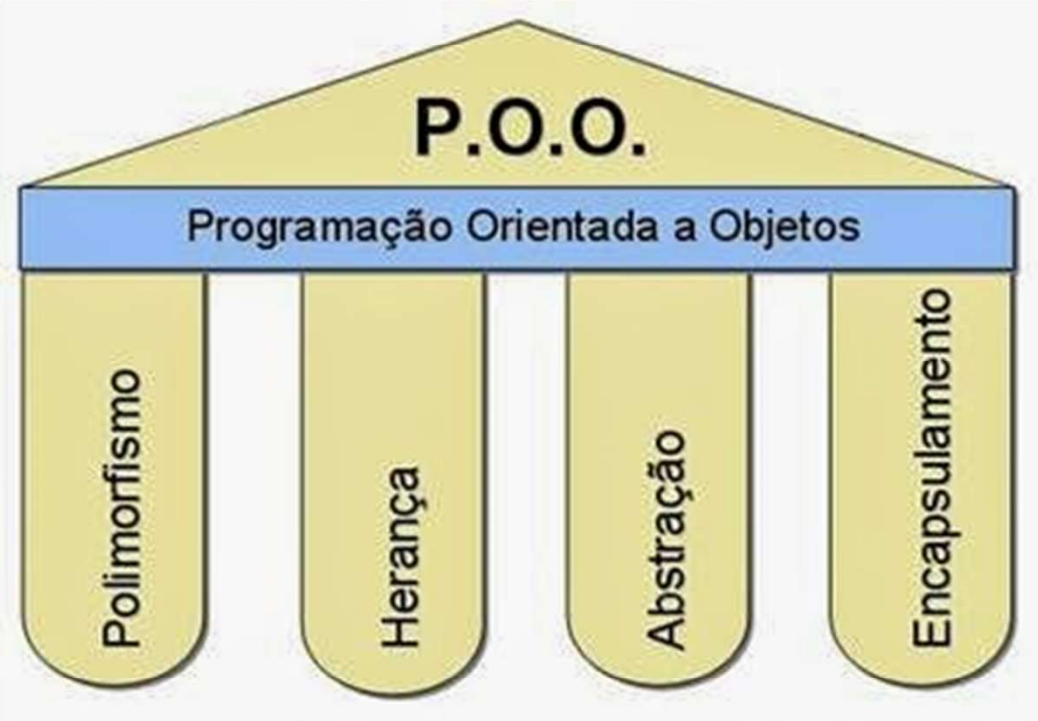
40 ANOS

Sejam bem-vindos!

Programação de Soluções Computacionais

2023.1

Programação Orientada a Objetos



Programação Orientada a Objetos

- É uma tecnologia que enxerga os sistemas como sendo coleção de objetos integrantes, permitindo melhorar a reusabilidade e extensibilidade dos softwares (FARINELLI, 2007).

- Ela engloba os princípios da abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência.

- Objetos são formas de representar um determinado elemento do mundo real (FARNIELLI, 2007), sendo instâncias de classes, que determina qual informação ele contém e como é possível manipulá-la.

Programação Orientada a Objetos

Exemplos de objetos com um cenário – Um cachorro

- Ele possui algumas características (que são chamadas de atributos), como:
  - Um nome;
  - Uma idade;
  - Um comprimento de pelos;
  - Cor dos pelos;
  - Cor dos olhos;
  - Um peso.

Programação Orientada a Objetos

Exemplos de objetos com um cenário – Um cachorro

- É possível, ainda, identificar um conjunto de ações:
  - Late;
  - Baba;
  - Corre;
  - Come;
  - Bebe;
  - Dorme...

Essas ações são chamadas de métodos/serviços/funções.

- Por exemplo: para alimentar o cachorro, utilizamos o método "comer".

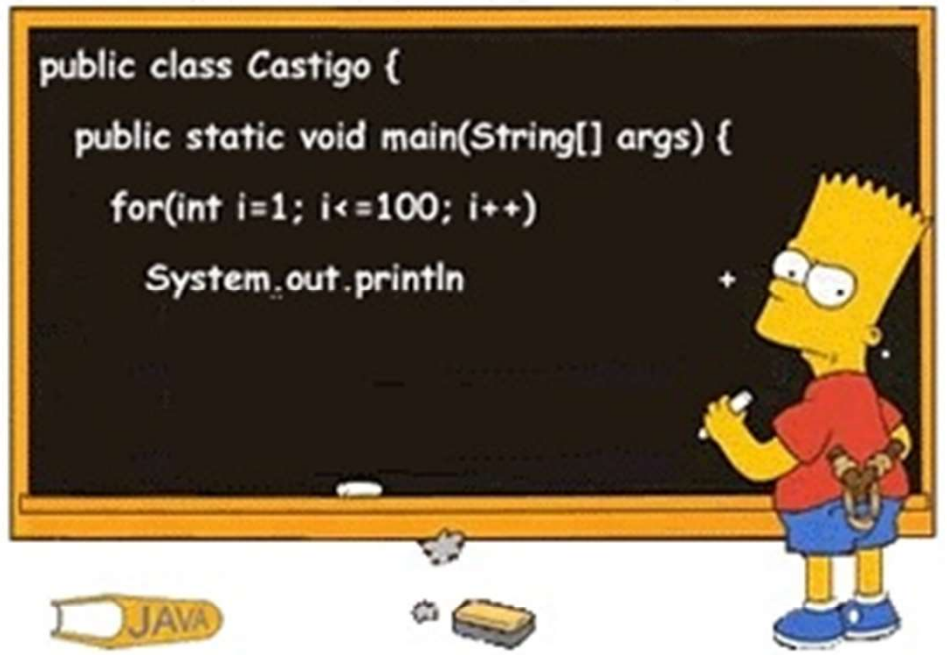
Programação Orientada a Objetos

- Atributos são variáveis ou campos que armazenam os valores que as características dos objetos podem conter (FARINELLI, 2007).
- De acordo com FARINELLI (2007),
  - “o estado de um objeto é o conjunto de valores de seus atributos, em um determinado instante. O comportamento de um objeto é como ele age e reage em termos de suas mudanças de estado e troca de mensagens com outros objetos“.

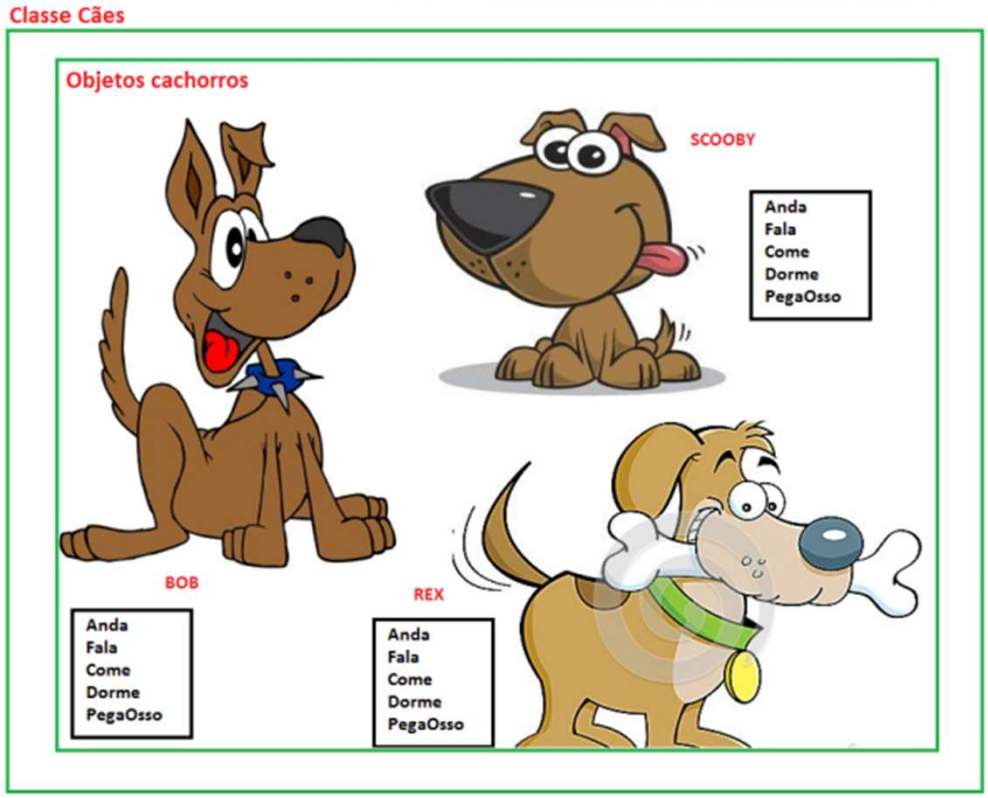
Cachorro	
Nome	Billy
Idade	12
Cor dos pelos	Marrom claro
Peso	10kg

Programação Orientada a Objetos – Classe

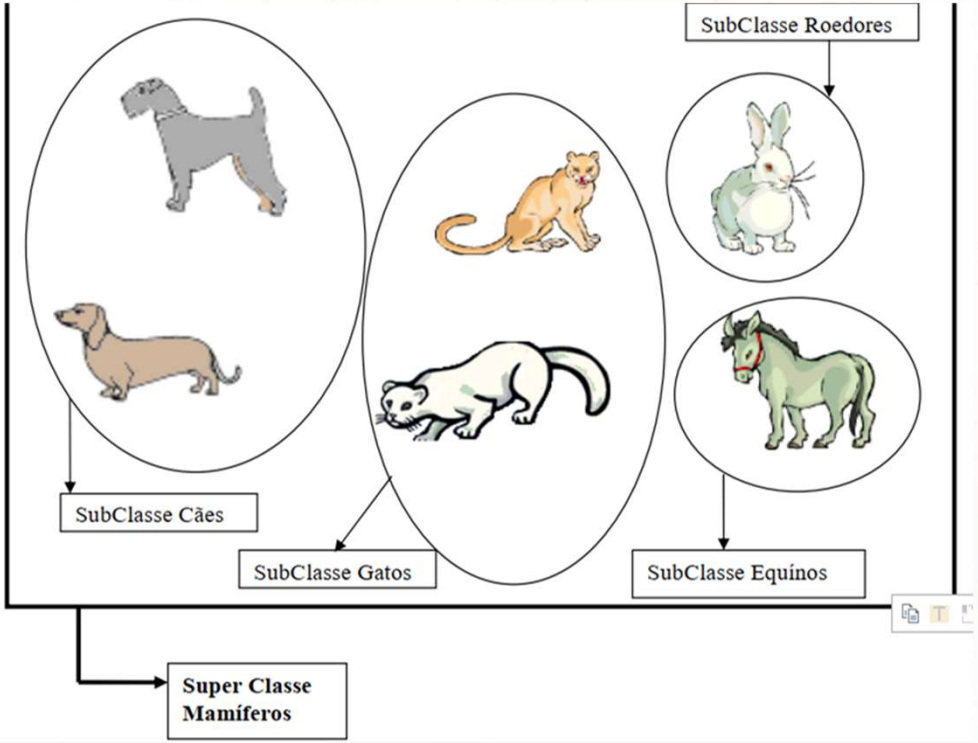
- Uma classe representa um conjunto de objetos que possuem características e comportamentos comuns.
- Um objeto é uma instância de uma determinada classe, ou seja, são baseados nas características definidas nas classes.

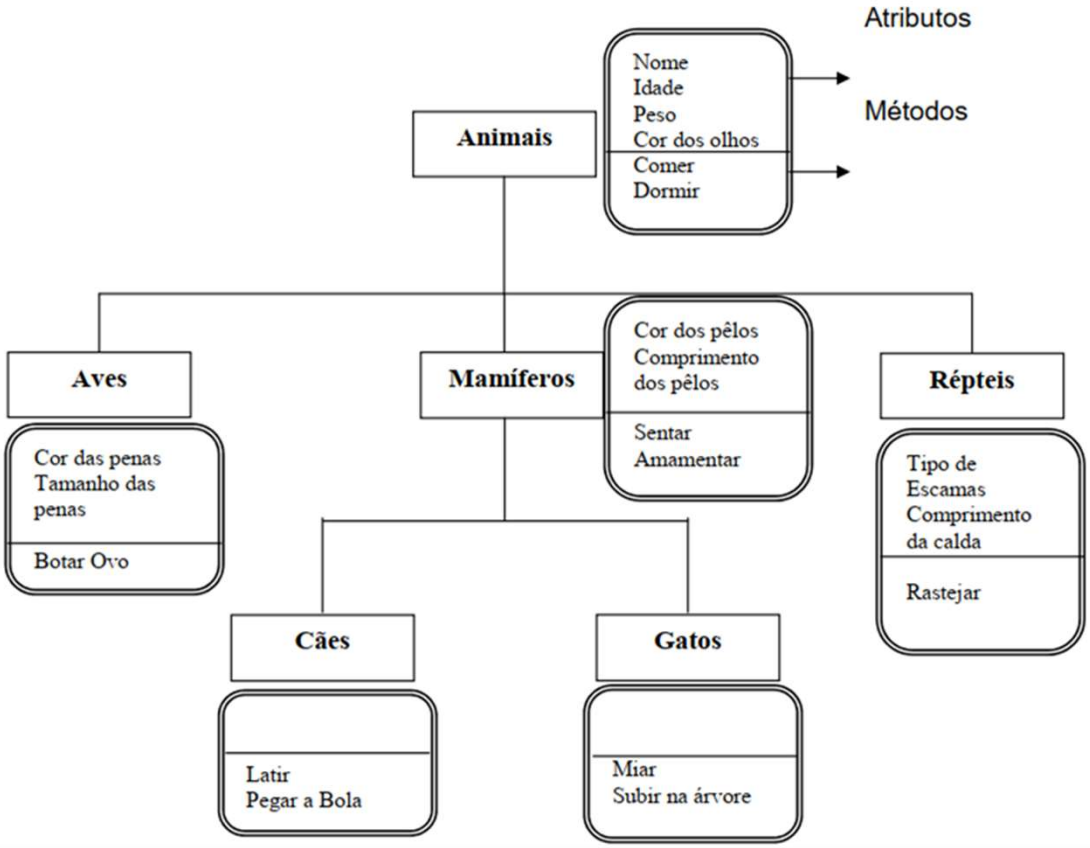


Programação Orientada a Objetos – Classe



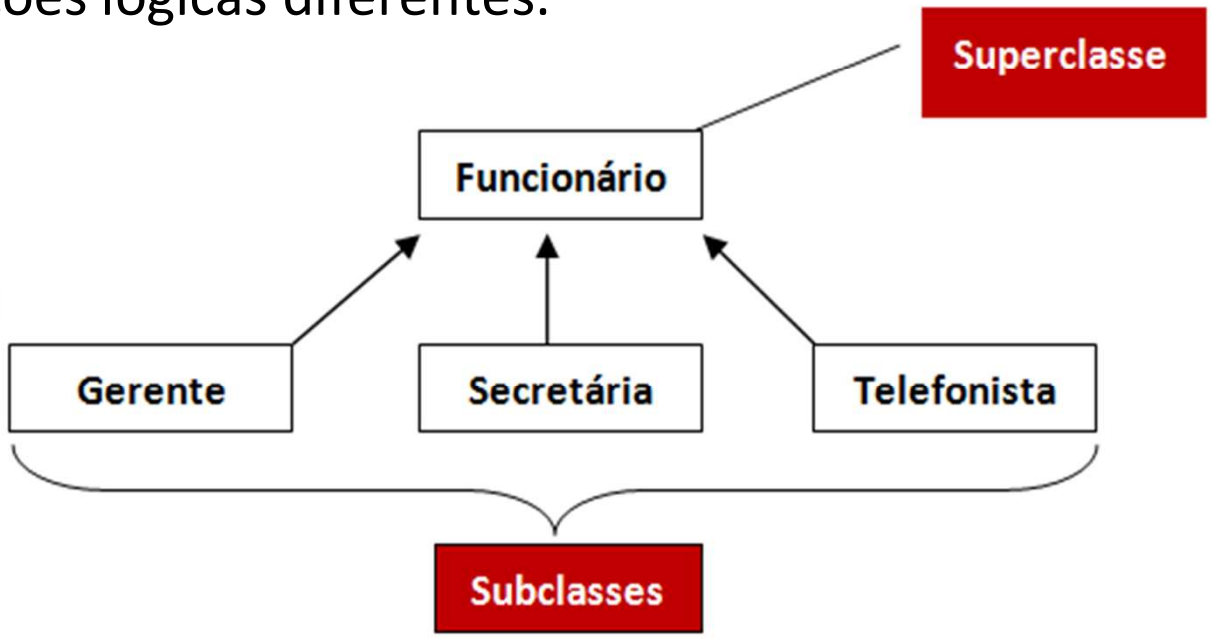
Programação Orientada a Objetos – Classe





Programação Orientada a Objetos

- Herança → uma maneira de reutilizar código a medida que podemos aproveitar os atributos e métodos de classes já existentes para gerar novas classes mais específicas que aproveitarão os recursos da classe hierarquicamente superior.
- Polimorfismo → os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.



Programação Orientada a Objetos – Herança

Professor

- nome
- email
- cpf
- idade
- sexo
- salário
- disciplina

Aluno

- nome
- email
- cpf
- idade
- sexo
- matrícula
- notas

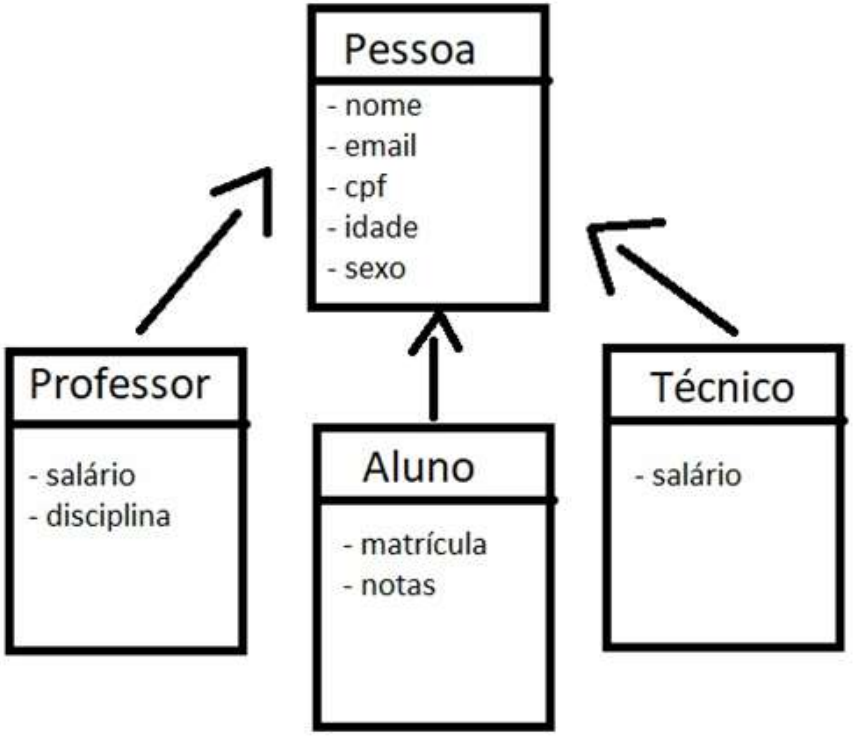
Técnico

- nome
- email
- cpf
- idade
- sexo
- salário

Programação Orientada a Objetos – Herança

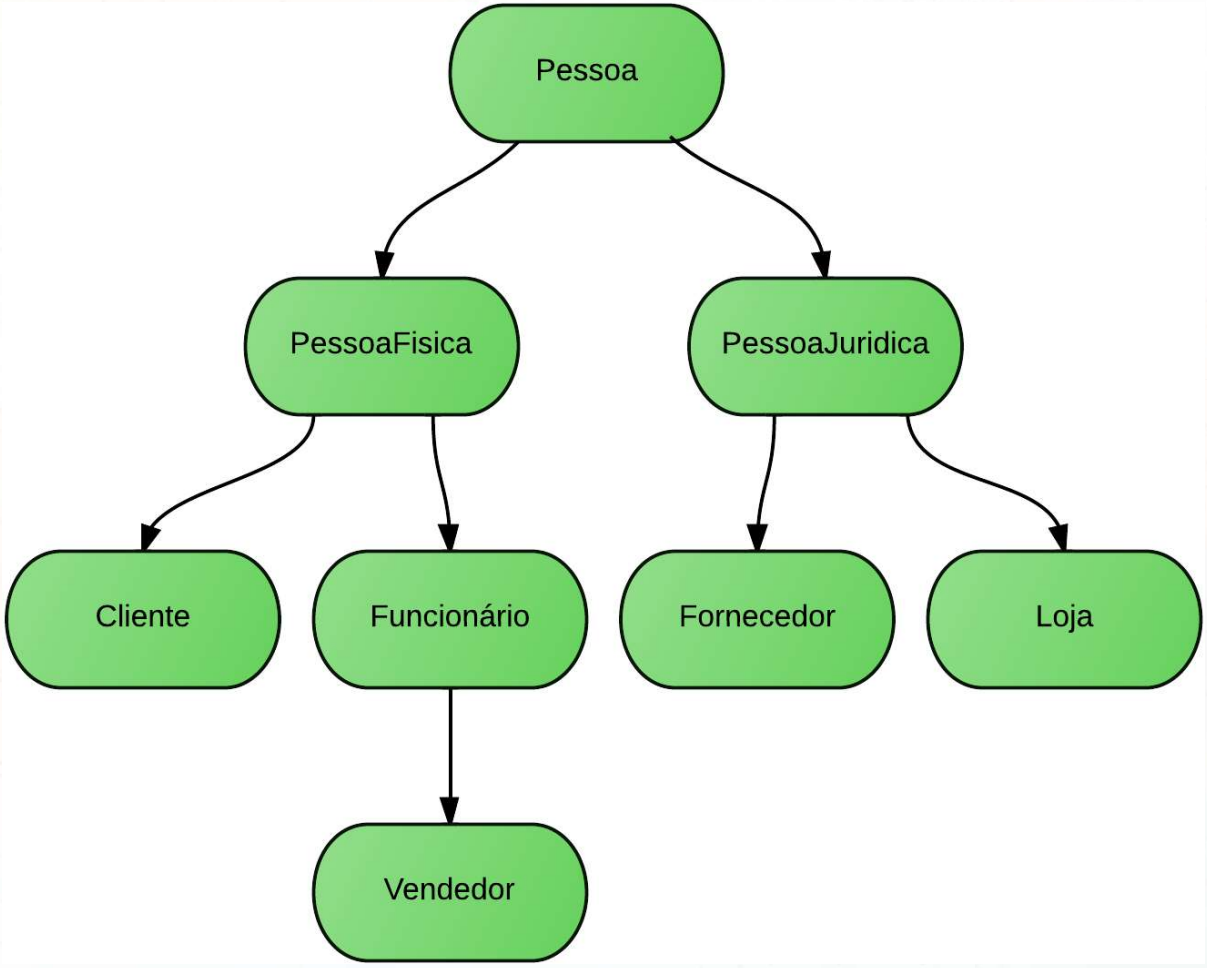
- A herança é um princípio da programação orientada a objetos (POO) que permite criar uma nova classe a partir de uma já existente.
- São criadas subclasses que contém atributos e métodos da classe primária (da qual deriva).
- A principal vantagem da herança é a capacidade para definir novos atributos e métodos para a subclasse, que se somam aos atributos e métodos herdados.

Programação Orientada a Objetos – Herança



Sempre que criarmos um objeto do tipo Professor, Aluno ou Técnico, este também possuirá os atributos e métodos da classe Pessoa.

Programação Orientada a Objetos – Herança



Programação Orientada a Objetos

- Abstração → é utilizada para a definição de entidades do mundo real, tendo como consideração as suas características e ações. Sendo onde são criadas as classes.

Entidade	Características	Ações
Carro, Moto	tamanho, cor, peso, altura	acelerar, parar, ligar, desligar
Elevador	tamanho, peso máximo	subir, descer, escolher andar
Conta Banco	saldo, limite, número	depositar, sacar, ver extrato

- Encapsulamento → é a técnica utilizada para não expor detalhes internos para o usuário, tornando partes do sistema mais independentes possível.

Métodos getters	Métodos setters

public String getNome() {
return nome;
}
	public void setNome(String nome) {
	this.nome = nome;
	}

public double getSalario() {
return salario;
}
	public void setSalario(double salario) {
	this.salario = salario;
	}

POO – Criando classes e aplicando as definições de herança, polimorfismo, encapsulamento e abstração

Programação Orientada a Objetos – Instância

- Uma instância é um objeto criado com base em uma classe definida;
- Classe é apenas uma estrutura, que especifica objetos, mas que não pode ser utilizada diretamente;
- Instância representa o objeto concretizado a partir de uma classe;
- Uma instância possui um ciclo de vida:
  - Criada → Manipulada → Destruída.

ă Programação Orientada a Objetos – Construtor e Método

- Construtor → Determina que ações devem ser executadas no momento da criação de um objeto (pode possuir ou não parâmetros).
  - Quando usamos a palavra chave ***new***, estamos construindo um objeto.
  - Sempre quando o ***new*** é chamado, ele executa o **construtor da classe**.
  - O construtor da classe é um bloco declarado com o **mesmo nome** que a classe.

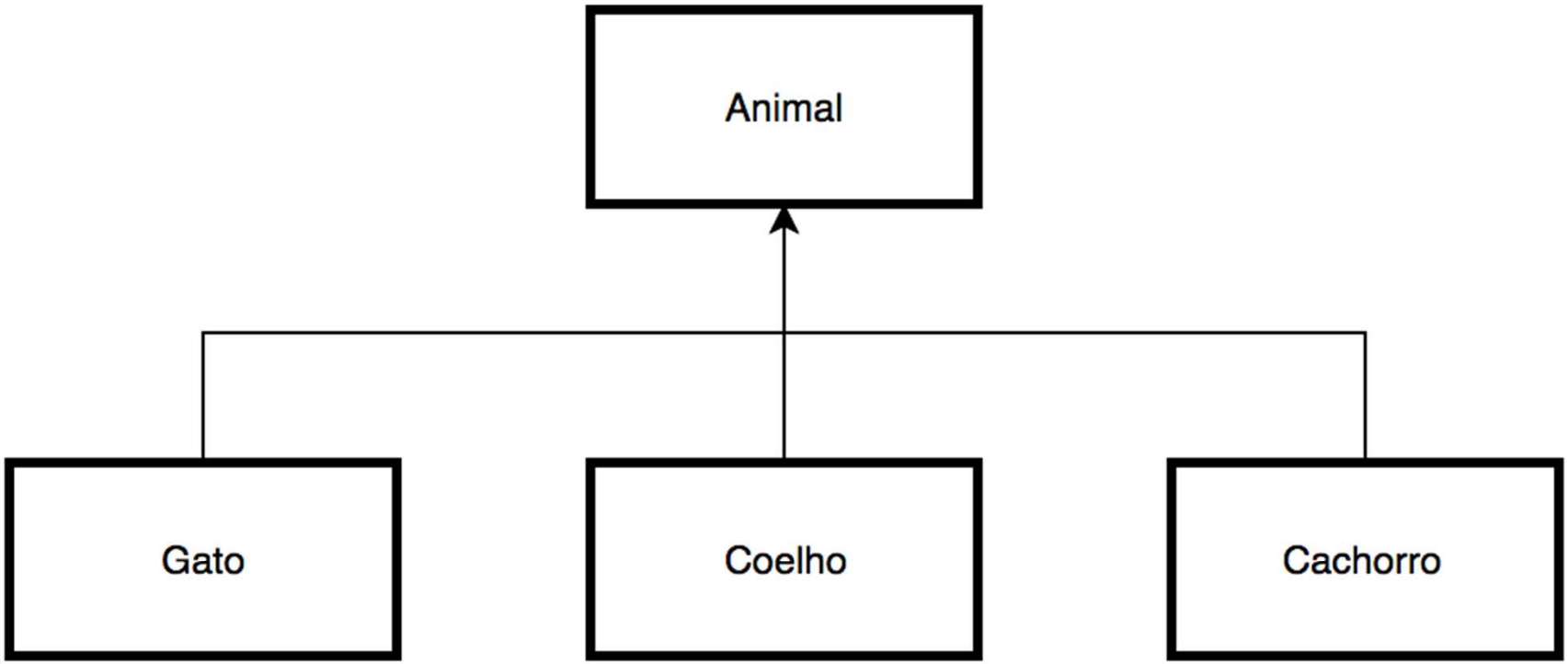
- Métodos → Representam os comportamentos de uma classe;
  - Permitem que acessemos os atributos, tanto para recuperar os valores, como para alterá-los caso necessário;
  - Podem retornar, ou não (void), algum valor;
  - Podem possuir, ou não, parâmetros.

Programação Orientada a Objetos

- Encapsulamento → Consiste em separar os aspectos externos de um objeto dos detalhes internos de implementação;
  - Evita que dados específicos de uma aplicação possa ser acessado diretamente;
  - Protege os atributos ou métodos de uma classe.

- Polimorfismo → É a capacidade de um objeto poder ser referenciado de várias formas. Significa "Muitas formas".
  - É a habilidade de uma entidade receber um objeto gerado a partir de uma subclasse e tratá-lo de forma genérica, como se fosse um objeto da superclassse.

Programação Orientada a Objetos – Atividade



Complementar

Programação Orientada a Objetos – Get's and Set's

- Para acessarmos os atributos de uma classe, a melhor forma é utilizando métodos.
- Os métodos GET e SET são técnicas para gerenciamento sobre o acesso dos atributos.
- Na criação dos métodos para acesso a esses atributos privados devemos colocar GET ou SET antes do nome do atributo.

Programação Orientada a Objetos – Método Get

- Acesso de atributos na classe;
- Retorno de valores.

```
public String getNomeFuncionario() {
        return nomefuncionario;
}

public int getMatricula() {
        return matricula;
}

public double getSalario () {
        return salario;
}
```

Programação Orientada a Objetos – Método Set

- Alterar/modificar valores de um atributo;
- Não tem um retorno, criando um método de tipo VOID.

```
public void setNomeFuncionario(Stringnomefuncionario) {
//  a utilização do THIS faz referência ao atributo da classe trabalhada.
// ex: este nome deste funcionário
        this.nomefuncionario = nomefuncionario;
}

public void setMatricula(int matricula) {
        this.Matricula = matricula;
}

public void setSalario (String salario) {
        this.salario= salario;
}
```

Programação Orientada a Objetos – Declaração de modificadores de acesso

- **public**
  - Pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.
- **private**
  - Os membros da classe não podem ser acessados ou usados por nenhuma outra classe;
  - Se aplica somente para seus métodos e atributos.
- **protected**
  - Torna o membro acessível às classes do mesmo pacote ou através de herança;
  - Seus membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados.

Programação Orientada a Objetos – Declaração de modificadores de acesso

- **default**
  - A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote;
- **final**
  - Não permite estende-la;
  - Não pode ser alterado depois que já tenha sido atribuído um valor.
- **abstract**
  - Não é aplicado nas variáveis, apenas nas classes;
  - Não pode ser instanciada.
- **static**
  - Usado para a criação de uma variável que poderá ser acessada por todas as instâncias de objetos da classe.

Programação Orientada a Objetos – Declaração de modificadores de acesso

	private	default	protected	public
mesma classe	sim	sim	sim	sim
mesmo pacote	não	sim	sim	sim
pacotes diferentes (subclasses)	não	não	sim	sim
pacotes diferentes (sem subclasses)	não	não	não	sim

Programação Orientada a Objetos – Encapsulamento

- O que vai controlar as proteções.
- Encapsulamento vem de encapsular, ou seja, separar o programa em partes, o mais isolado possível.
- Serve para controlar o acesso aos atributos e métodos de uma classe.
- É uma forma eficiente de proteger os dados manipulados dentro da classe.

Programação Orientada a Objetos – Encapsulamento

```
public class Carro{
   private int capacidadeTanque;
   private int qtdeCombustivel;

   public Carro(int capacidadeTanque){
        this.capacidadeTanque = capacidadeTanque;
   }

   public void abastecer(int qtdeCombustivel){
     if((this.qtdeCombustivel + qtdeCombustivel) > this.capacidadeCombustivel){
       System.out.println("Quantidade excede o limite do tanque!");
      }
      else{
       this.qtdeCombustivel = this.qtdeCombustivel + qtdeCombustivel;,
       System.out.println("Carro abastecido. Quantidade atual: " + this.qtdeCombustivel );
      }
   }
}
```

Retirado de: PINHEIRO, Rafael. 2016.

Programação Orientada a Objetos – Encapsulamento

```
Carro c = new Carro(50);
c.abastecer(40); // Imprime: "Carro abastecido. Quantidade atual: 40"
c.abastecer(15); // Imprime: "Quantidade excede o limite do tanque"
c.abastecer(10); // Imprime: "Carro abastecido. Quantidade atual: 50"
```

Externamente não é possível saber que o método **abastecer** está acessando os atributos **capacidadeTanque** e **qtdeCombustivel** e por vezes alterando esse segundo.

Retirado de: PINHEIRO, Rafael. 2016.

Referência:

- FARINELLI, Fernanda. **Conceitos básicos de programação orientada a objetos**. 2007.