SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

Pilares POO

Desenvolvimento de Sistemas

Prof. Me. Reneilson Santos

Março/2024

SENAI
PELO FUTURO DO TRABALHO

Agenda

- ➔ Abstração
- ➔ Encapsulamento
- ➔ Herança
- ➔ Polimorfismo

Pilares da Programação Orientada a Objetos

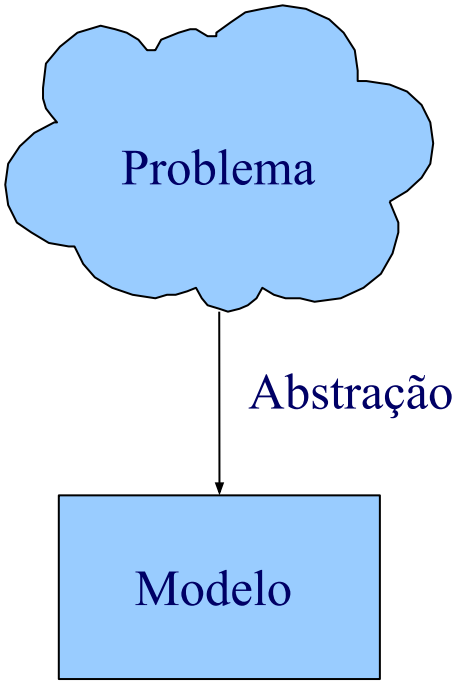
O paradigma de programação orientada a objetos possui 4 pilares que formam a sua estrutura e que definem uma linguagem orientada a objetos, são eles:

1. Abstração
2. Encapsulamento
3. Polimorfismo
4. Herança

Abstração

Abstração

- Abstração é um modo de pensamento em que nos concentramos em idéias gerais em vez de manifestações específicas dessas idéias
- É a disciplina em que nos concentramos em aspectos essenciais do problema e ignoramos todos os outros aspectos não essenciais.



O modelo define uma vista abstrata do problema

Exemplo

- Sistema de administração de estudantes:
  - Detalhes importantes:
    - Nome,
    - data de nascimento,
    - CPF
  - Detalhes irrelevantes:
    - Altura,
    - cor dos cabelos,
    - Peso,
    - hobbies

Abstração em Java

Abstração em Java

A abstração em Java permite que você modele e represente conceitos complexos de maneira mais simples e eficaz.

Ela promove a reutilização de código, facilita a manutenção e ajuda a criar sistemas mais organizados e flexíveis.

Encapsulamento

Encapsulamento

- Ocultamento da informação é uma prática pela qual o projetista se **restringe** à interface pública de um tipo para propósitos de inspeção ou modificação.
- Um tipo abstrato de dados é um tipo acrescido com a noção de ocultamento da informação.
- Ocultamento da informação proporciona um **nível de proteção contra acessos inesperados** à estrutura de dados.

Encapsulamento em Java

Encapsulamento em Java

Em Java, o encapsulamento é alcançado principalmente através dos seguintes conceitos:

- Modificadores de acesso (private, public, protected);
- Liberando acesso apenas para as funcionalidades necessárias (deixando-os público).

```java
public class ContaBancaria {
    private double saldo;

    public void depositar(double valor) {
        saldo += valor;
    }

    public double consultarSaldo() {
        return saldo;
    }
}
```

Herança

Herança

A Herança possibilita que as classes **compartilhem** seus **atributos**, **métodos** e outros elementos da classe entre si.

Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente, definindo assim dois tipos de classes:

- Classe base (classe pai), que concede características a uma outra classe;
- Classe derivada (classes filhas), que “herda” as características da classe base;

Herança em Java

Java possui três tipos de itens que permitem a realização de herança:

- Classes genéricas
- Classes abstratas

Sendo que classes abstratas são usadas especificamente para herança, enquanto as classes genéricas podem ser usadas na herança mas a princípio não são criadas com este intuito.

Herança em Java - Classe Base

```
public class Pessoa {
    public String nome;
    public String cpf;
    public Date dataNascimento;

    public Pessoa(String nome, String cpf, Date data)
{
        this.nome = nome;
        this.cpf = cpf;
        this.dataNascimento = data;
    }
}
```

Herança em Java - Classes que Herdam

```
public class Funcionario extends Pessoa {
    public Funcionario(String nome, String cpf, Date data) {
        super(nome, cpf, data);
    }
    public double salario;
    public Date dataAdmissao;
    public String cargo;
}
```

Classes Abstratas

Uma classe abstrata é uma classe que não pode ser instanciada diretamente. **Ela serve como um modelo para outras classes que herdam dela**.

Uma classe abstrata pode conter métodos abstratos (métodos sem implementação) e métodos concretos (métodos com implementação).

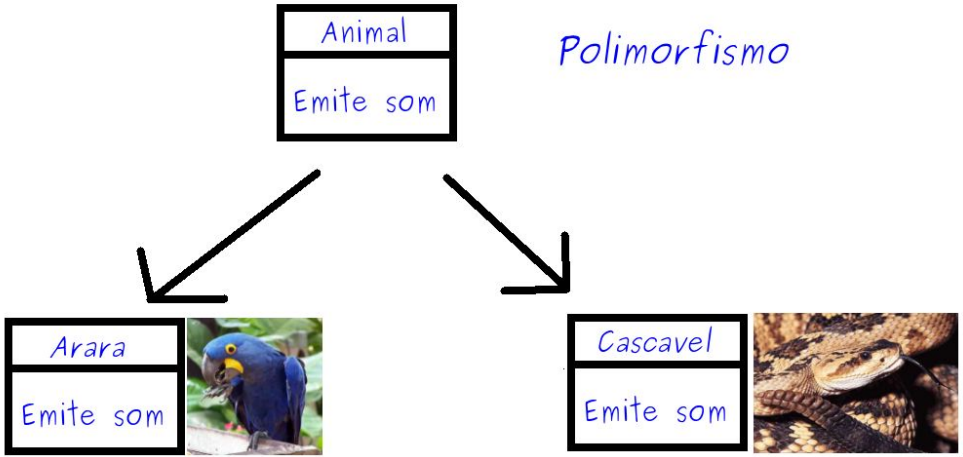
As classes derivadas (subclasses) que herdam de uma classe abstrata **devem implementar os métodos abstratos** definidos na classe abstrata.

```
abstract class Forma {
    abstract double calcularArea(); // Método abstrato, sem implementação
}

class Circulo extends Forma {
    double raio;

    Circulo(double raio) {
        this.raio = raio;
    }

    @Override
    double calcularArea() {
        return Math.PI * raio * raio;
    }
}
```

Polimorfismo

Polimorfismo

Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.



Polimorfismo em Java

Polimorfismo - Java

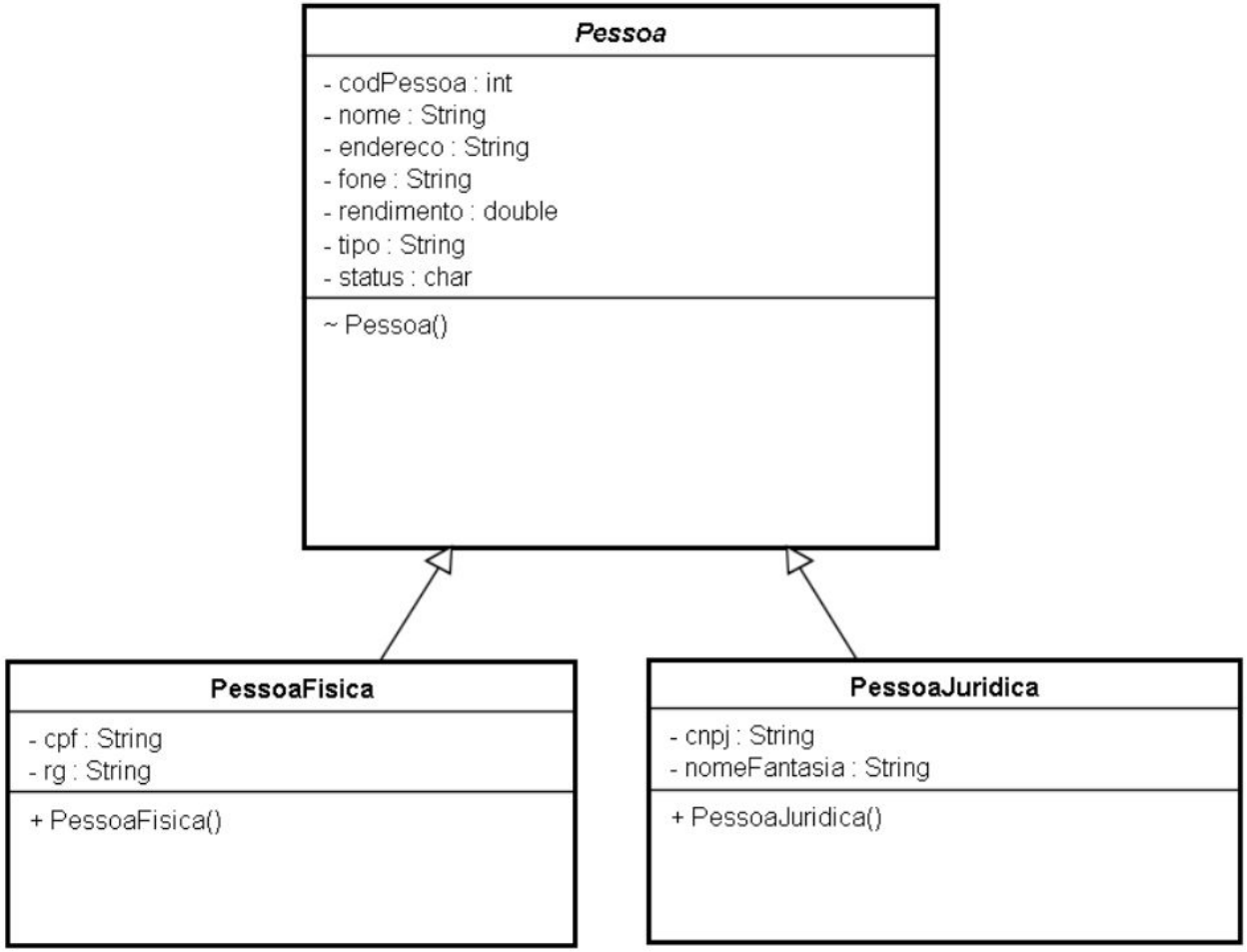
Como já vimos, ao implementar métodos que são de classes herdadas, em Java podemos adicionar a anotação ***@Override***, os métodos com essa anotação são métodos onde estão implementados polimorfismo, pois, são métodos que estão sobrescrevendo alguma funcionalidade prévia daquele objeto e nós estamos substituindo por uma nova dentro daquela classe.

Pode ser alcançado por meio de sobrecarga de métodos ou implementação de interfaces.

```java
public interface Animal {
    void fazerSom();
}

public class Cachorro implements Animal {
    public void fazerSom() {
        System.out.println("Au au!");
    }
}

public class Gato implements Animal {
    public void fazerSom() {
        System.out.println("Miau!");
    }
}
```

Atividade

Implementar o diagrama de classes (criando o método construtor, gets e sets).


Pessoa
- codPessoa : int
- nome : String
- endereco : String
- fone : String
- rendimento : double
- tipo : String
- status : char
~ Pessoa()
PessoaFisica
- cpf : String
- rg : String
+ PessoaFisica()
PessoaJuridica
- cnpj : String
- nomeFantasia : String
+ PessoaJuridica()


Kahoot!