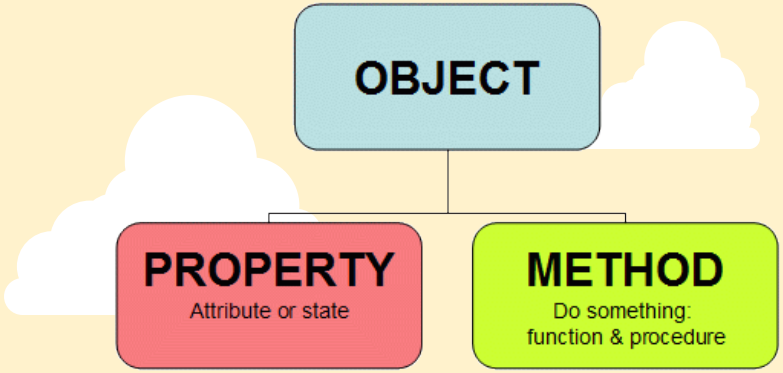
Programação Orientada por Objetos



Sistemas de Informação

Prof. Fábio L. R. Cordeiro

SUMÁRIO

- Motivação
- Paradigmas
- Estruturada vs POO
- Conceitos POO
- Objetos, atributos, Classes
- Os Pilares da POO
- Abstração, Encapsulamento,
- Herança, Polimorfismo
- Vantagens

MOTIVAÇÃO

1) Quais os problemas com a Programação Estruturada

2) Por que mudar a maneira de programar "quebrando! Um paradigma aceito e utilizada há decadas

MOTIVAÇÃO

1) Quais os problemas com a Programação Estruturada

2) Por que mudar a maneira de programar "quebrando! Um paradigma aceito e utilizada há decadas

"COMPLEXIDADE"

Motivação

Capacidade Humana

Mercury is the closesA limitação da capacidade humana para compreender um sistema como um todo.

Sistemas Complexos

Sistemas complexos, grandes com um conjunto vasto de comportamentos, com grande ciclo de vida além de muitos usuários.

Manutenção

Dificuldade de manutenção, pelo tamanho do programa aliado a quantidade de usuários, fez com que o problema da **quantidade** e **diversidade** das aplicações requeressem novas maneiras para se programar de forma mais produtiva e **sustentável**.

Paradigma

Conjunto de diretrizes que regem como resolver um problema de determinada natureza.

- Os paradigmas são capazes de influenciar a percepção, auxiliando-nos na organização, coordenação e na maneira como olhamos para o mundo.
- Em Programação de Sistemas, dizemos que:
  - Os paradigmas determinam como são estruturados os elementos que compõem uma programa e como são dadas suas interações entre si e com agentes externos.

Programação Estruturada vs Orientada a Objetos

PE

Estruturada

Modelagem dos dados com base no conceito de módulos ou subprogramas.

Exemplos

C; Fortran; Cobol; Natural; PL/SQL.

POO

Orientada a Objetos

Modelagem com base no conceito de Classes e seus Relacionamentos.

Exemplos

It's theC++; C#; Java; SmallTalk;

Programação
Estruturada vs Orientada a Objetos

PE

POO

Estruturada

Orientada a Objetos

Conceitos de Orientação a Objetos

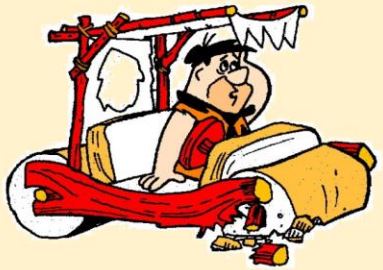
- No mundo real, tudo é definido com objetos, sejam concretos ou abstrados.
- Esses objetos se relacionam entre si e com o ambiente de várias maneiras.
- Exemplos:

Pessoa, Música, Guitarra, Alegria, Depressão...




Conceitos de Orientação a Objetos

Para entender bem esse conceito de OO vamos nos perguntar:



Dos veículos nas figuras:

- O que existe de similar entre eles?
- O que existe de diferente?
- E se você fosse avaliá-los, qual vale mais?

Objeto

- Na concepção de Sistemas de Informação, temos que um objeto é qualquer coisa existente no mundo real, concreto ou abstrato (conceitualmente):

  Como: Aluno, Professor, automóvel, disciplina, Livro,
  Janela do Windows, Botão, Caixa de Diálogo.

- A técnica de especificar os objetos abstraídos do mundo real para um programa, damos o nome de: **Modelagem**.

Atributos

- Os objetos possuem características ou propriedade chamadas de atributos que identificam o estado de um objeto.

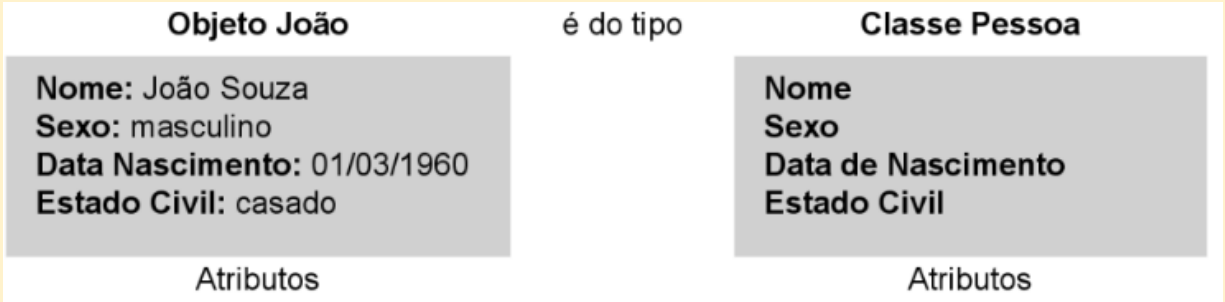
- Um atributo é uma abstração do tipo de dados ou estado que os objetos da classe possuem.

- Exemplo de um objeto da Classe Humano:
  Nome: Boba Fett
  Filiação: Jango Fett
  Gênero: Masculino
  Nascimento: 32 ABY
  Espécie: Humano (clone)
  Profissão: Mercenário
  Altura: 1,83

Classes

Quando são identificadas características e operações similares em objetos distintos realizando assim uma classificação:

Uma Classe é a representação de um conjunto de objetos que compartilham a mesma estrutura de atributos, operações e relacionamentos dentro de um mesmo contexto.



Exemplo de um objeto (instância) da classe Pessoa

Classes e Objetos

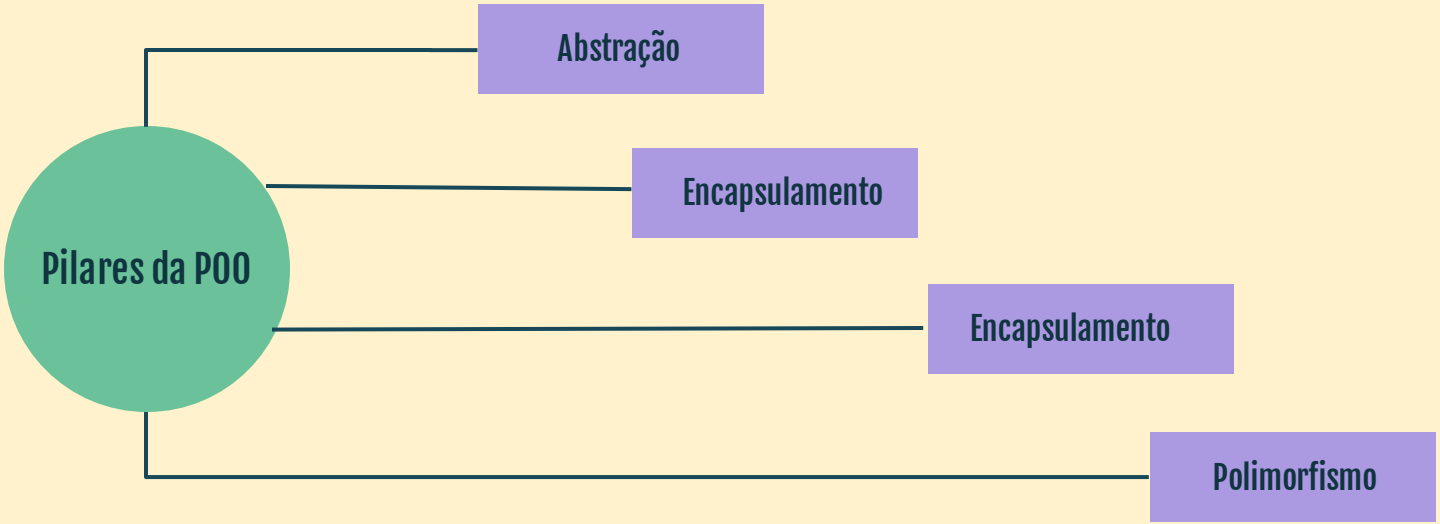
Classe é uma categoria de **entidades** afins.
Exemplos: Carro, Avião, Pessoa etc.

Objetos: uma entidade específica existente, uma **instância** de uma classe.

Exemplos:

PILARES DA POO

Para que uma linguagem seja caracterizada como Orientada a Objetos é necessário que ela atenda impreterivelmente os seguintes recursos:



Pilares da POO

Abstração

A abstração caracteriza-se pela imaginação das operações que serão realizadas por um objeto real dentro do sistema.

- **identidade**: garantia da unicidade do objeto dentro do sistema, ou seja, o que representa de forma única um objeto dentro de um sistema:
  Ex: Nomes exclusivos dentro de Pacotes ou *Namespaces*

- **Propriedades**: são as características que definem um objeto.
  Ex: o Objeto Carro pode ter: Cor, Modelo, Marca etc.

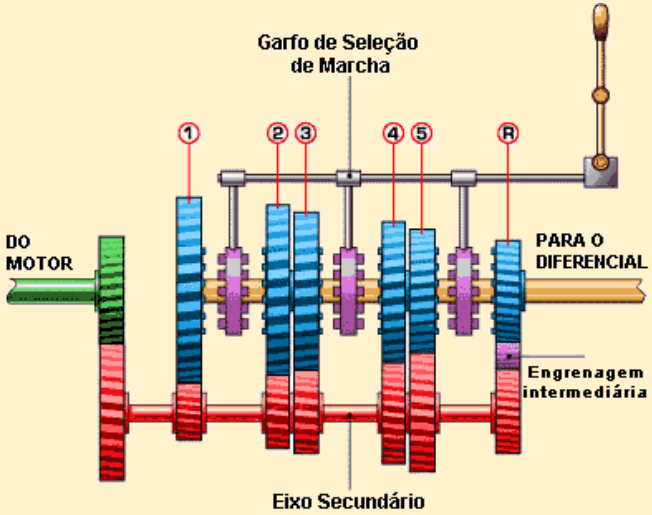
- **Métodos**: Definem as operações (eventos, ações) que um objeto pode executar.
  Ex: O objeto Carro pode ultrapassar( ), acelerar( ), bater( )

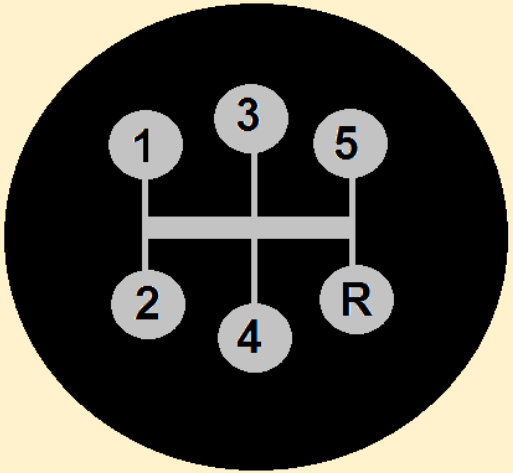
Pilares da POO

Encapsulamemto

Encapsulamento (*data hiding*)

- Exemplo: Na classe Carro, não é necessário saber como as engrenagens do câmbio funcionam internamente, porem você sabe através da interface da Alavanca de Marchas, onde colocar as marchas do carro.





Pilares da POO

Encapsulamemto

Encapsulamento (*data hiding*)

- É definido como uma técnica para minimizar as interdependências entre módulos, através da definição de interfaces externas.

- Como uma “caixa preta” – não é necessário que se saiba como funciona internamente uma determinada função, mas sim o método de utilizá-la

Exemplo: Na classe Carro, você não sabe como as engrenagens do câmbio do carro funciona internamente, porem você sabe através da interface da Alavanca de Marchas, onde colocar as marchas do carro.

Pilares da POO

Encapsulamemto

Segurança: Protege os objetos de terem seus atributos corrompidos por objetos.

Independência: “Esconder” atributos, assim um objeto protege outros de complicações de dependência da sua estrutura interna.

Pilares da POO

Herança

Como o nome diz, é a propriedade de objetos especialistas herdarem características de objetos generalistas.

- A capacidade provida pela Orientação por Objetos de facilitar a reutilização ou reuso de códigos é uma das grandes vantagens da POO.

Dependendo da linguagem utilizada pode ser vista em dois tipos:

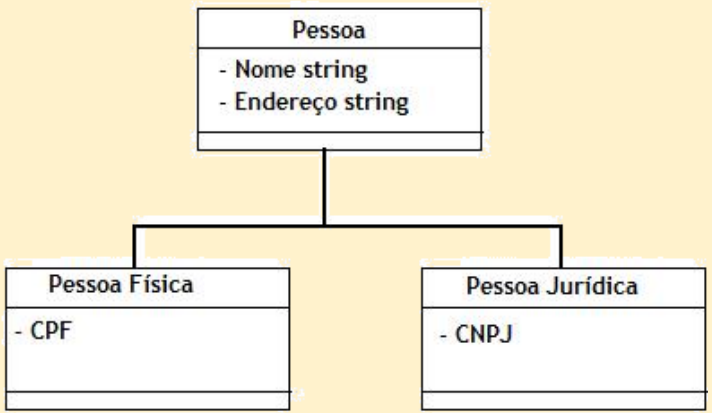
Herança Simples e **Herança Múltipla**

Pilares da POO

Herança

Herança Simples

Exemplo de Herança Simples, onde as Classes Filhas Pessoa Física e Jurídica, herdam os atributos Nome e Endereço da classe Mãe Pessoa.

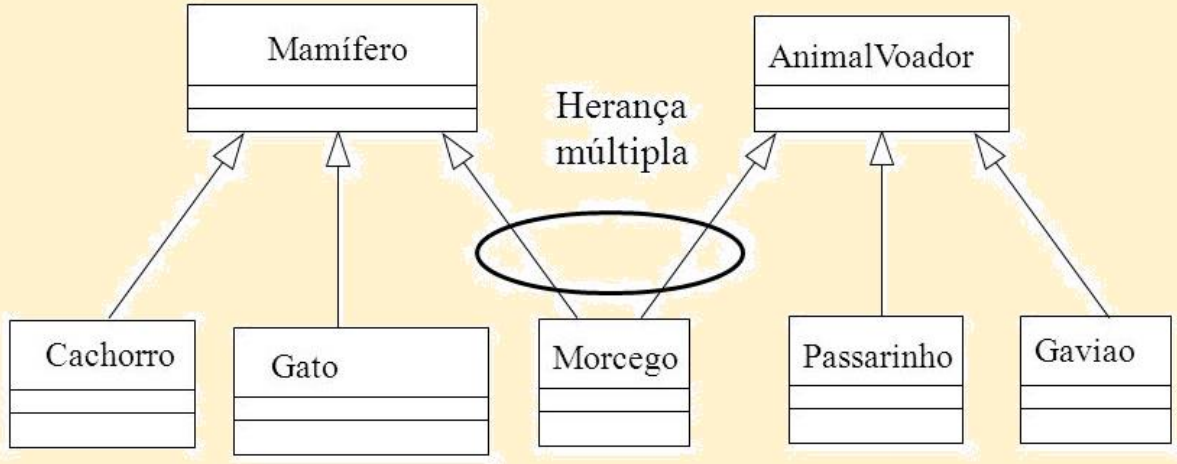


Pilares da POO

Herança

Herança Múltipla

Exemplo de Herança Múltipla, onde as Classe Morcego herda as características de duas classes Mãe ao mesmo tempo..



Polimorfismo

De maneira básica; **polimorfismo** é a capacidade que os objetos tem de **alterar** seu funcionamento interno de métodos herdados por um objeto Pai.

Cada linguagem implementa o Polimorfismo com determinadas peculiaridades.

Em JAVA, utiliza-se a notação **@override**

Em C# a palavra-chave **virtual**

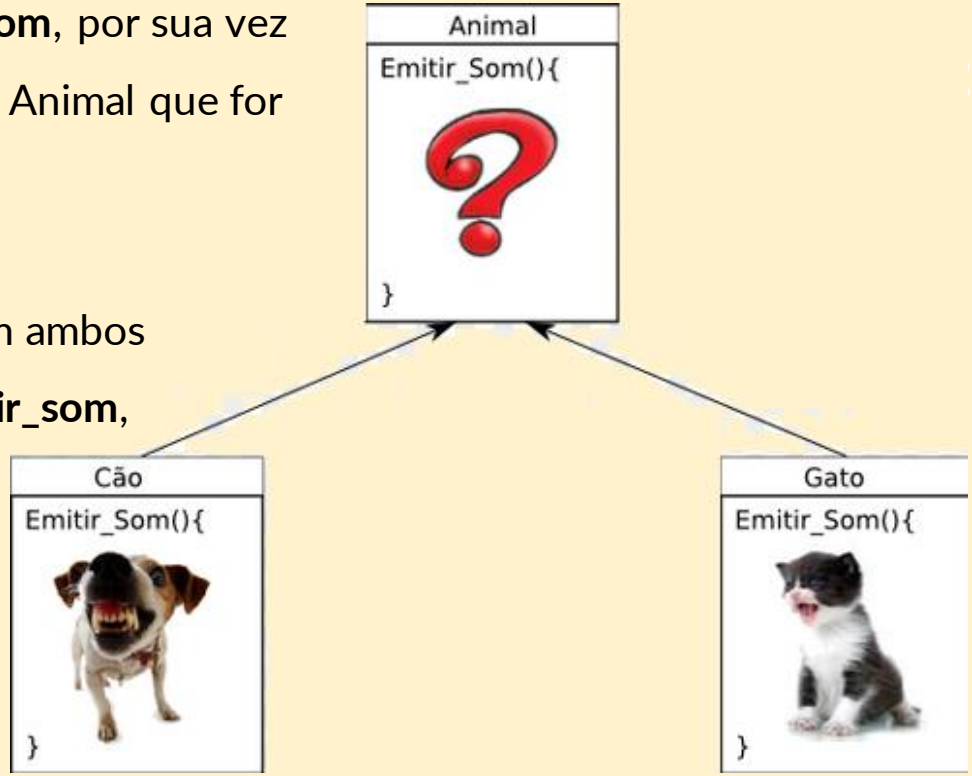
Ambos antes dos métodos que serão “sobrescritos”

Polimorfismo

Exemplo:

A classe Pai Animal tem o método **Emitir_Som**, por sua vez este método varia de acordo com o tipo do Animal que for herdar esse método.

A classe Cão: late, a classe Gato: mia, porem ambos são chamados com o mesmo método “**emitir_som**, que de acordo com a Classe instanciada vai ter um funcionamento diferente.



Vantagens do Polimorfismo

Torna mais produtiva as atividades de programação e manutenção de Aplicações.

Característica unificadora; tratando todas as etapas do desenvolvimento sob a mesma abordagem.

Reutilização

O processo de reutilização fica mais natural sem inventar a roda.

Escalabilidade

O sistema pode escalar e evoluir de acordo com a necessidade sem grandes problemas na estrutura

Manutenabilidade

Sistemas mais coesos e menos acoplados pormitem mais facilidade de manutenção

Referências

LOUREIRO, Henrique. **C# 5.0 com Visual Studio®** 2012: curso completo. Lisboa: FCA, c2013. xxii, 585 p. ISBN 9789727227525 (Disponível no Acervo).

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. **Algoritmos: lógica para desenvolvimento de programação de computadores**. 26. ed. rev. São Paulo: Érica, 2012. 328 p. ISBN 9788536502212 (Disponível no Acervo).

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos, c2010. xvi, 302 p. ISBN 9788521617501 (Disponível no Acervo).