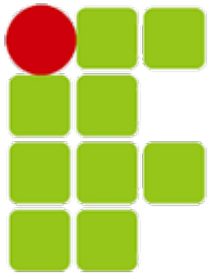


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAIANO CAMPUS GUANAMBI

ANÁLISE DE SISTEMAS

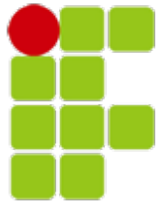
Orientação a Objetos

PROF. Igor Campos

Objetivos dessa aula

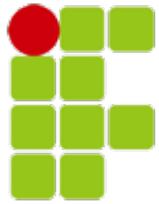
- ✓ Apresentar os conceitos sobre OO:
  - ✓ Origem
  - ✓ Conceito
  - ✓ Objetos
  - ✓ Atributos
  - ✓ Métodos
  - ✓ Classes
  - ✓ Herança
  - ✓ Visibilidade

Introdução



- Necessidade de abordagens para desenvolver software de maneira organizada e estruturada
  - Análise Estruturada
  - Análise Essencial
  - Análise OO
  - ...

Introdução



Programação Estruturada

- Base:
  - Sequência: Uma tarefa é executada após a outra, linearmente.
  - Decisão: A partir de um teste lógico, determinado trecho de código é executado, ou não.
  - Iteração: A partir de um teste lógico, determinado trecho de código é repetido por um número finito de vezes.
- Vantagens
  - É fácil de entender. Ainda muito usada em cursos introdutórios de programação.
  - Execução mais rápida.
- Desvantagens
  - Baixa reutilização de código
  - Códigos confusos: Dados misturados com comportamento

Técnicas de programação tradicionais

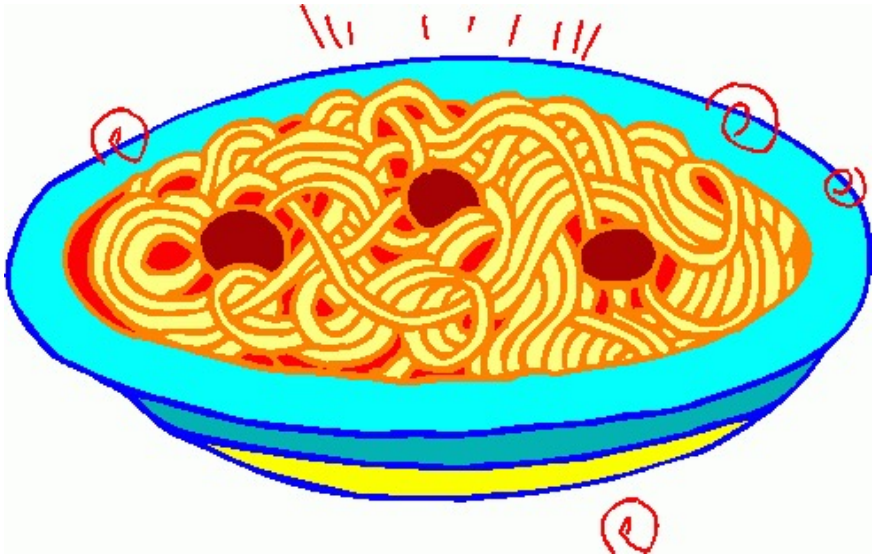
As técnicas de programação tradicionais, como por exemplo a "decomposição funcional", leva o desenvolvedor a decompor o sistema em partes menores (funções), criando um emaranhado de inúmeras funções que chamam umas às outras.

Geralmente não há separação de conceitos e responsabilidades, causando dependências enormes no sistema, dificultando futuras manutenções no código do programa.

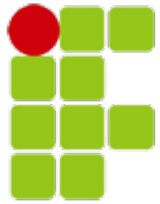
Não existe muito reaproveitamento de código, ao contrário, muitas vezes se tem muito código duplicado.

Técnicas de programação tradicionais

É a famosa programação espaguete...



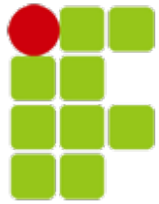
e pode causar uma grande indigestão.



Programação Orientada a Objetos

- Base
  - Classes e Objetos
  - Métodos e Atributos
- Vantagens
  - Melhor organização do código
  - Bom reaproveitamento de código
- Desvantagens
  - Desempenho mais baixo que o paradigma estruturado
  - Mais difícil compreensão

Orientação a Objetos



- O paradigma da Orientação a Objetos, ou Programação Orientada a Objetos (POO ou OOP), eleva a programação e o desenvolvimento de sistemas para um novo patamar.

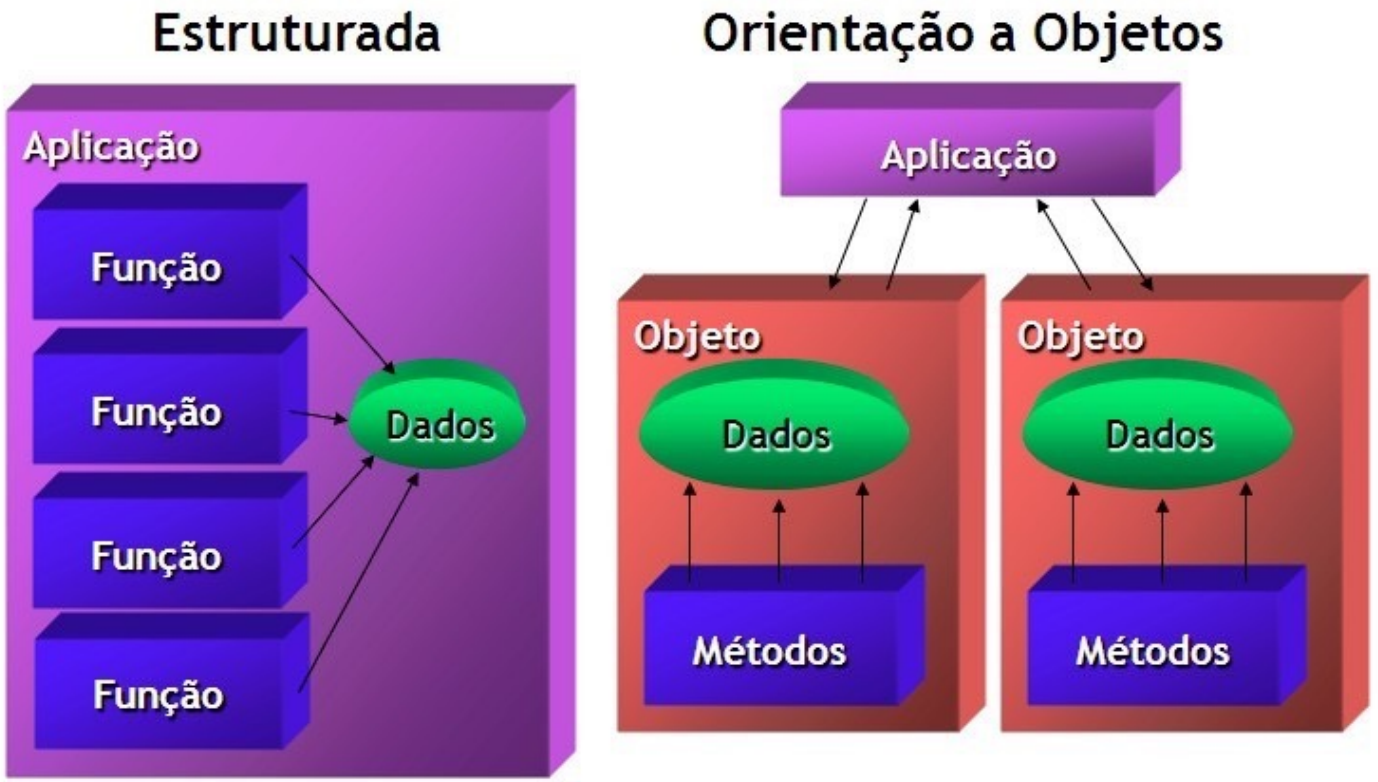
- A OO é um mecanismo moderno que ajuda a definir a estrutura de programas baseada nos conceitos do mundo real, sejam eles reais ou abstratos.

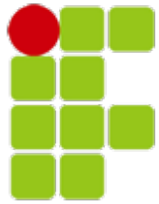
- A OO permite criar programas componentizados, separando as partes do sistema por responsabilidades e fazendo com que essas partes se comuniquem entre sí, por meio de mensagens.

- Essas partes do sistemas são chamadas de OBJETOS.

Programação Orientada a Objetos

Programação Orientada a Objetos vs Estruturada





O que é OO ?

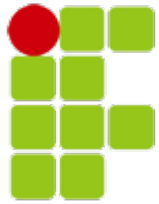
Conceitos Básicos

- Orientação a Objetos (OO) é um método de programação (paradigma) que usa tipo de dados personalizados.
- Em vez de operar apenas com tipo de dados primitivo (int, double.., podemos construir novos tipo de dados.

- Os objetos se comunicam a parti da toca de mensagens.
- Mensagem é um sinal enviado de um objeto ao outro, requisitando um serviço, usando uma operação programada no objeto chamado.

- Essas mensagens resultam na ativação de métodos, os quais realizam as ações necessárias.

Conceitos Básicos



- Os objetos que compartilham uma mesma interface, ou seja, respondem as mesmas mensagens, são agrupados em classes.

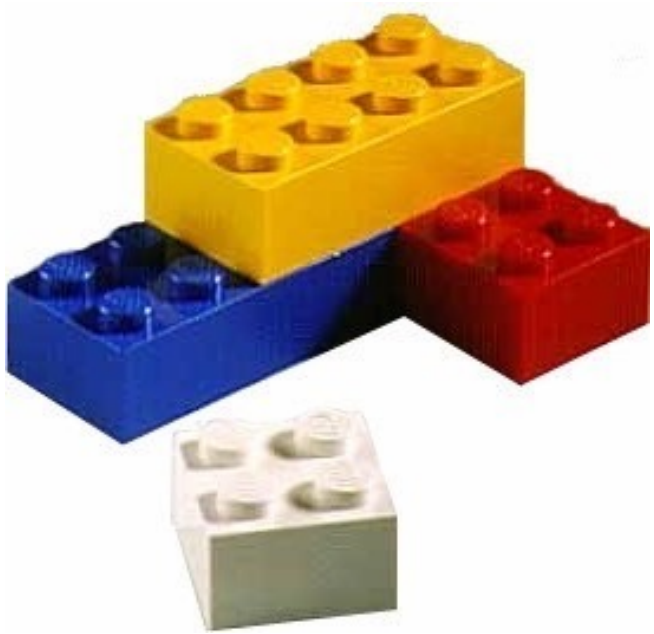
- Objeto é algo DINÂMICO: é criado por alguém, tem uma vida, e morre ou é morto por alguém. Assim, durante a execução do sistema, os objetos podem:
  - ser construídos
  - executar ações
  - ser destruídos
  - tornar-se inacessíveis


Orientação a Objetos
Encapsulamento
Herança
Composição
Polimofismo
Princípio da abstração

Orientação a Objetos

A OO introduz e enfatiza os seguintes conceitos:

- Objeto
- Mensagem
- Classe
- Abstração



Objeto

Objetos são a chave para se compreender a tecnologia orientada a objetos. Você olha ao seu redor e tudo o que vê são objetos: carro, mesa, janela, livro, pessoa, etc.

Os objetos do mundo real têm duas carecterísticas em comum: **ESTADO** e **COMPORTAMENTO**.

Estado

O estado de um objeto revela seus dados importantes. Por exemplo, uma pessoa tem: idade, peso, altura, cor de cabelo, cor da pele.

Comportamento

O comportamento são as ações que aquele objeto pode exercer ou executar. Por exemplo, uma pessoa pode: andar, falar, ouvir, pular.

Objeto

Esses objetos podem ser tanto objetos concretos (carro, livro, nota fiscal), quanto conceitos abstratos (conta corrente, venda, pessoa jurídica).

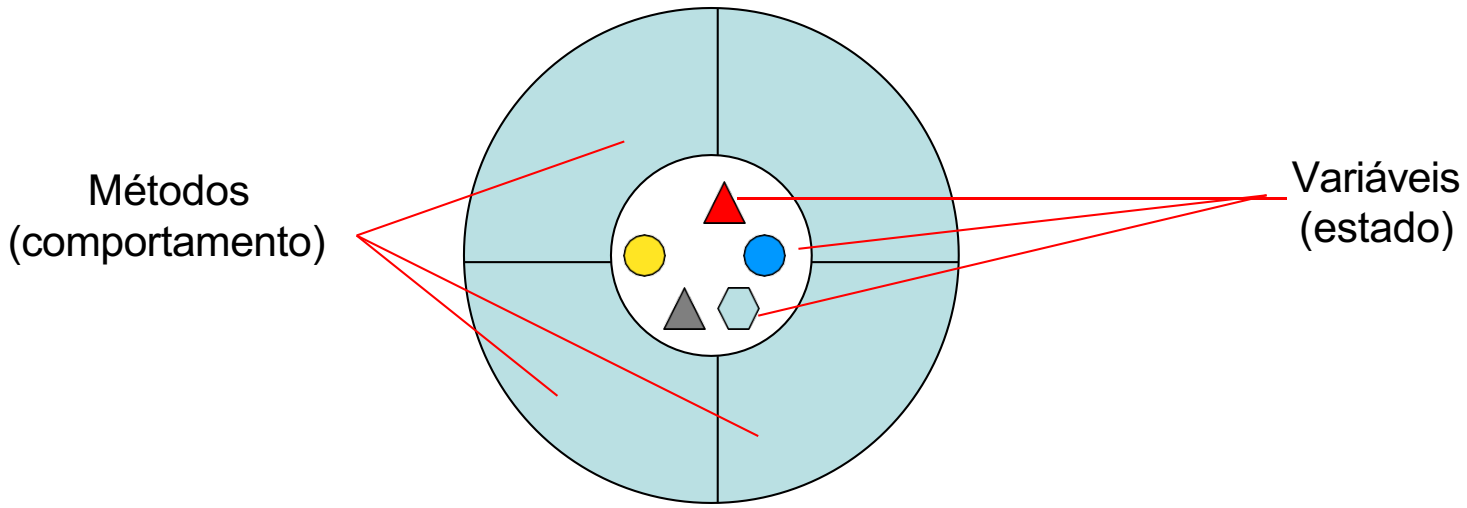
Na Orientação a Objetos, os objetos do mundo real são modelados e representados no mundo computacional, ou seja, dentro do sistema, por meio de objetos de software.

Cada objeto deve ser conhecido, bem definido e ter seu limite e um significado dentro do sistema.

Objeto

Um objeto de software mantém seu estado em uma ou mais de suas variáveis. Ele implementa seu comportamento através de seus métodos. Método é o mesmo que função ou procedimento.

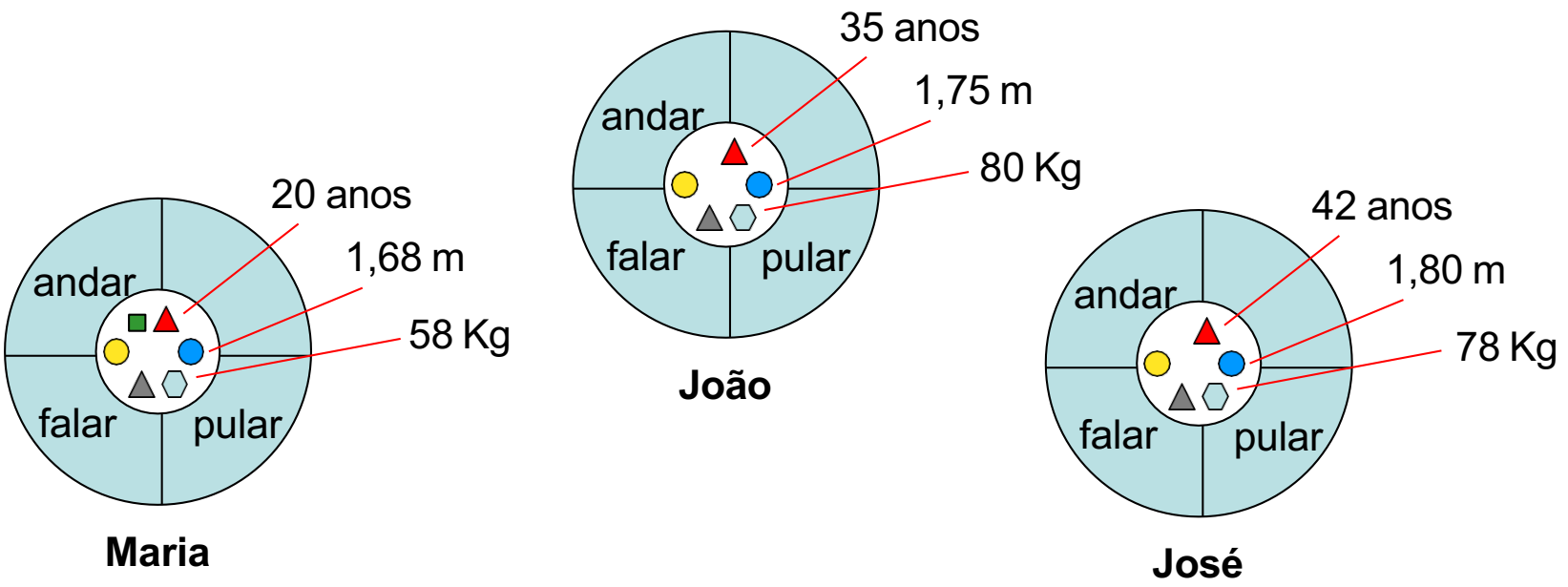
Por definição: Um objeto é um pedaço de software que possui variáveis (estado) e métodos (comportamento).



Objeto

Um sistema pode conter um ou inúmeros objetos ativos. Cada objeto ativo no sistema em particular também é chamado de instância. As diferentes instâncias possuem seu próprio estado.

O exemplo abaixo mostra várias intâncias de pessoas.

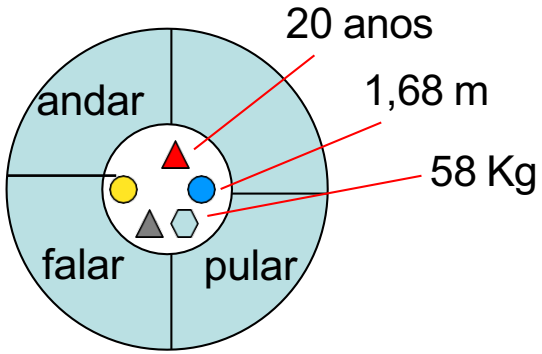


Objeto

Cada instância de pessoa possui um estado diferente em particular, como visto na última figura.

Porém, cada instância, além do estado, também possui seus métodos (comportamento) que operam sobre o próprio estado. Em outras palavras, para pular, cada pessoa vai fazer uma determinada força dependendo da sua idade, altura e peso, por exemplo.

A idéia é que cada objeto seja responsável por seus dados (estado) e seja capaz de realizar as próprias operações que lhe foram atribuídas (comportamento).

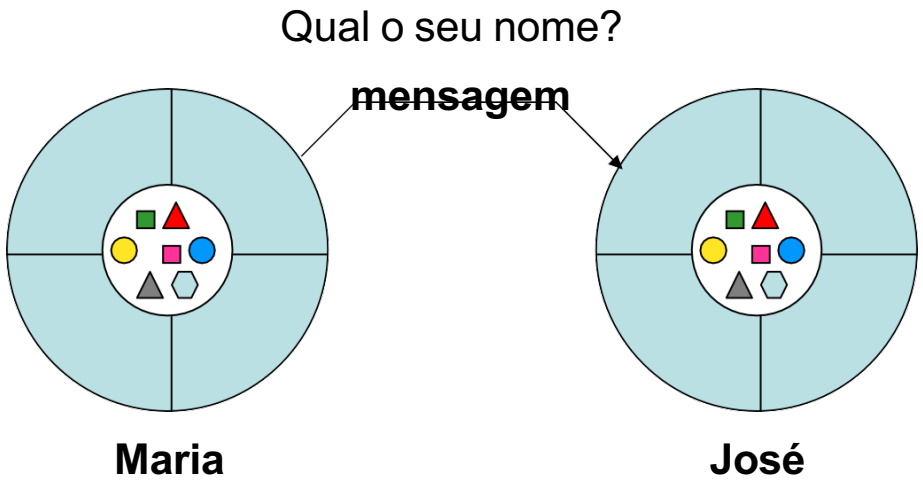


Maria

Mensagem

Um objeto por si só não significa muito em um sistema. Para ter algum sentido e valor esses objetos precisam interagir e comunicar-se entre si.

Os objetos se comunicam por meio de mensagens.



Quando um objeto **A** quer se comunicar com um objeto **B** é enviada uma mensagem de **A** para **B**.

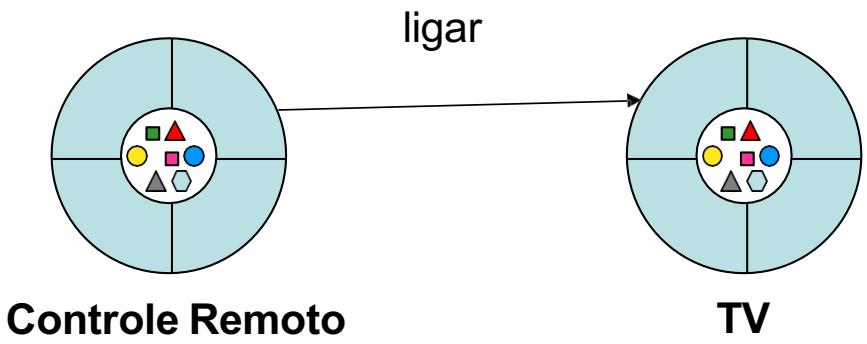
Mensagem

Enviar uma mensagem significa executar um método.

Então, se **A** envia uma mensagem para **B**, podemos entender como o objeto **A** executando um método do objeto **B**.

As mensagens são compostas por três partes:

- Objeto a quem a mensagem é endereçada
- Nome do método a ser chamado
- Parâmetros que o método recebe



Classe

No mundo real frequentemente percebemos vários objetos de um mesmo tipo. Por exemplo: seu carro é um dos muitos carros existentes no mundo.

Usando a terminologia OO, dizemos que um carro em particular é uma instância da classe de objetos conhecida como carros.

Os carros, em geral, possuem estado (cor, potência do motor, combustível) e comportamento (ligar, acelerar, brecar, mudar marcha) em comum.

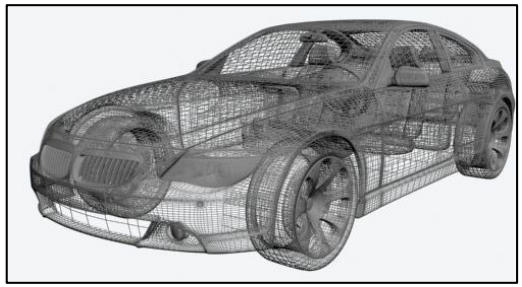
O estado de cada carro é independente e pode ser diferente do estado dos outros carros. Cada carro pode ter uma cor diferente, por exemplo.

Classe

Na Orientação a Objetos também é possível ter vários objetos do mesmo tipo, que compartilham caracteríscas em comum.

Tirando vantagem dessa semelhança entre alguns objetos, também é possível criar modelos para esses objetos. Esse modelo é chamado de **CLASSE**. As classes são tipos que podem ser criados.

Por definição: Uma classe é um modelo (protótipo) que define as variáveis (estado) e os métodos (comportamento) comuns a todos os objetos do mesmo tipo.



Classe

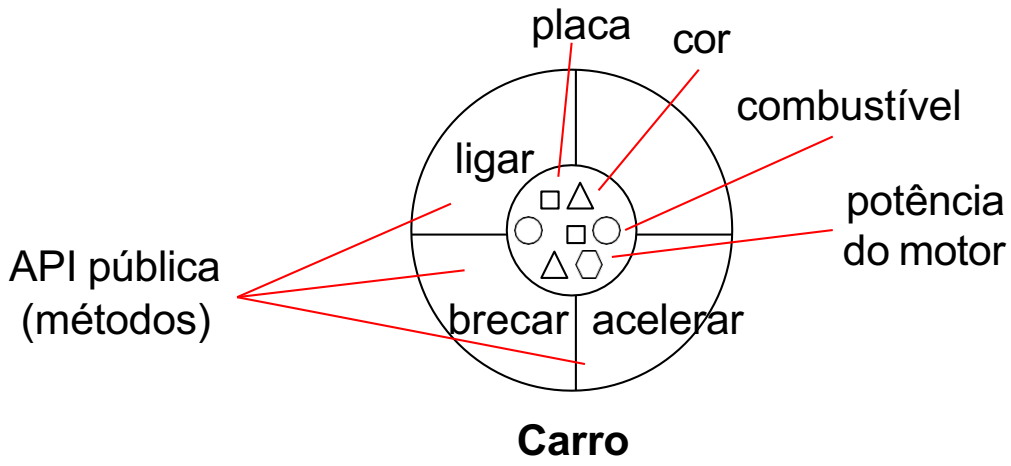
Objeto

Classe

Na classe são definidas as variáveis e implementados os métodos.

Os objetos são criados a partir de suas classes.

A cada objeto criado o sistema aloca memória para o novo objeto e suas variáveis.



Carro

Comumente fazem confusão entre classes e objetos. Lembre-se que classe define as características comuns e os objetos são instâncias dessas classes, com estado próprio.

Classe x Objeto

Gato

New

Gato
Raça: Savannah
Nome: Gatuno
Peso: 2,5 quilos
Idade: 2 anos

New

Gato
Raça: Maine Moon
Nome: Listrado
Peso: 3 quilos
Idade: 5 anos

New

Gato
Raça: Siamês
Nome: Bichano
Peso: 4 quilos
Idade: 3 anos

Atributos

Representam um conjunto de informações, ou seja, elementos de dados que caracterizam um objeto.

Descrevem as informações que ficam escondidas em um objeto para serem exclusivamente manipuladas pelas operações daquele objeto.

São variáveis que definem o estado de um objeto, ou seja, são entidades que caracterizam os objetos.

Cada objeto possui seu próprio conjunto de atributos

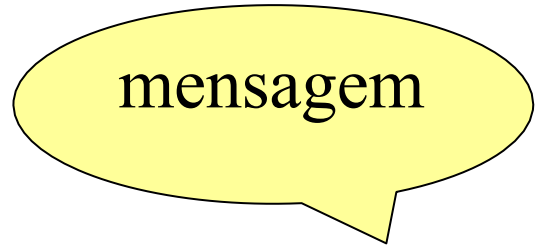
Métodos

São procedimentos definidos e declarados que atuam sobre um objeto ou sobre uma classe de objetos

Métodos são invocados por Mensagens

Cada objeto possui seu próprio conjunto de métodos

Métodos X Mensagem



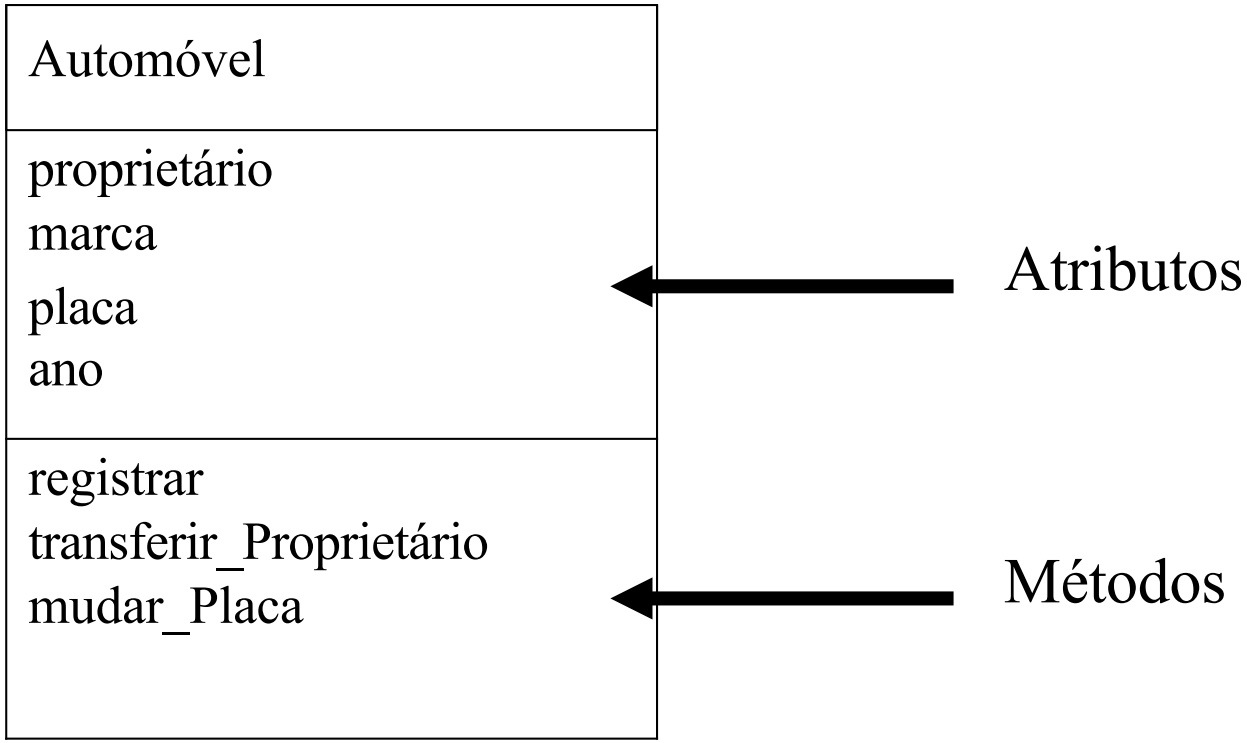

```
le1.alterarNome('Rosa Olivera')
```

le1: Leitor

nome = Maria dos Santos
numeroUsp = 342343
dataNascimento = 04/25/1973


```
método alterarNome(Char[30] novoNome)
Inicio
    nome := novoNome;
Fim
```

Atributos e Métodos



Exercício

- Cite 4 atributos de um aluno
  - 1
  - 2
  - 3
  - 4
- Cite 3 métodos de um aluno
  - 1
  - 2
  - 3

Herança

Herança é um mecanismo da OO que permite criar novas classes a partir de classes já existentes, aproveitando-se das características existentes na classe a ser extendida.

Este mecanismo é muito interessante pois promove um grande reuso e reaproveitamento de código existente.

Com a herança é possível criar classes derivadas (subclasses) a partir de classes bases (superclasses). As subclasses são mais especializadas do que as suas superclasses, mais genéricas.

As suclasses herdam todas as características de suas superclasses, como suas variáveis e métodos.

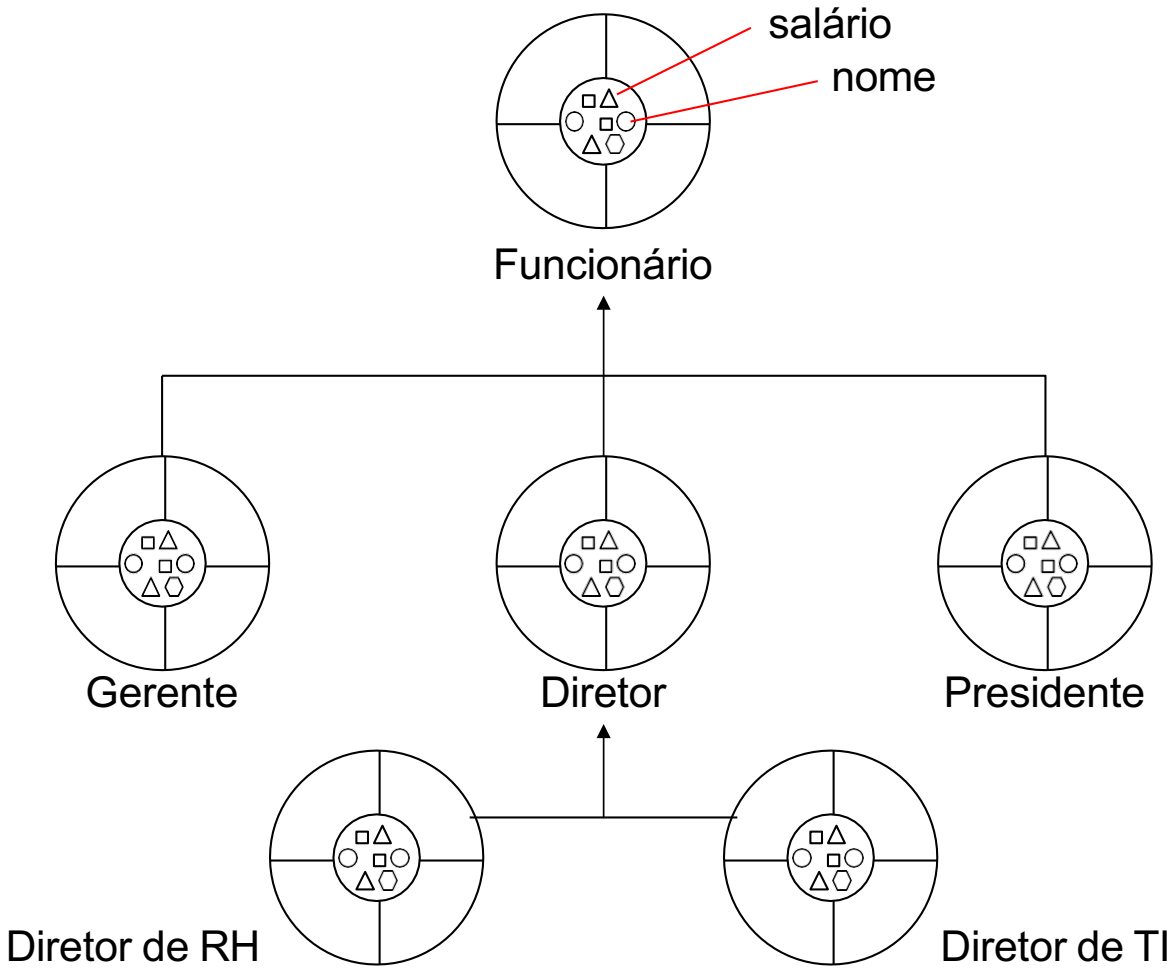
Herança

Imagine que dentro de uma organização empresarial, o sistema de RH tenha que trabalhar com os diferentes níveis hierárquicos da empresa, desde o funcionário de baixo escalão até o seu presidente.

Todos são funcionários da empresa, porém cada um com um cargo diferente. Mesmo a secretária, o pessoal da limpeza, o diretor e o presidente possuem um número de identificação, além de salário e outras características em comum.

Essas características em comum podem ser reunidas em um tipo de classe em comum, e cada nível da hierarquia ser tratado como um novo tipo, mas aproveitando-se dos tipos já criados, através da herança.

Herança


salário
nome
Funcionário
Gerente
Diretor
Presidente
Diretor de RH
Diretor de TI

Herança

Os subtipos, além de herdarem todas as características de seus supertipos, também podem adicionar mais características, seja na forma de variáveis e/ou métodos adicionais, bem como reescrever métodos já existentes na superclasse (polimorfismo).

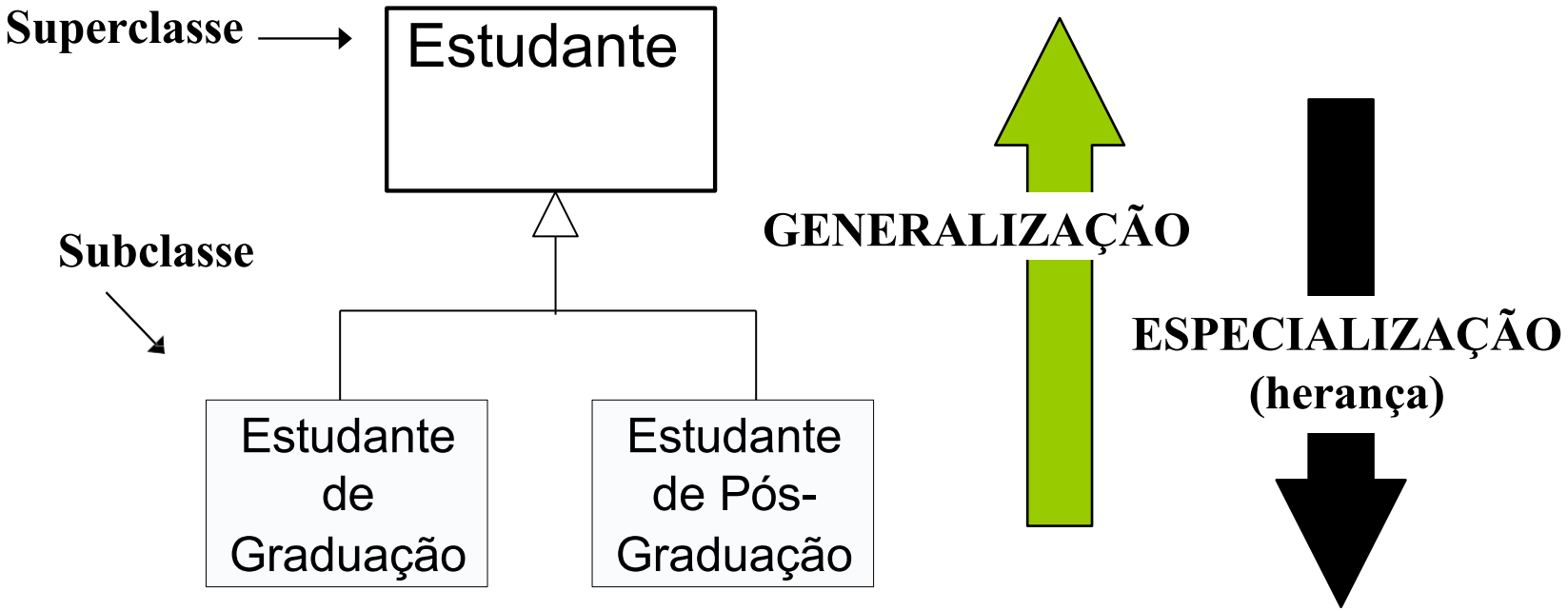
A herança permite vários níveis na hierarquia de classes, podendo criar tantos subtipos quanto necessário, até se chegar no nível de especialização desejado.

Podemos tratar subtipos como se fossem seus supertipos, por exemplo o sistema de RH pode tratar uma instância de Presidente como se fosse um objeto do tipo Funcionário, em determinada funcionalidade.

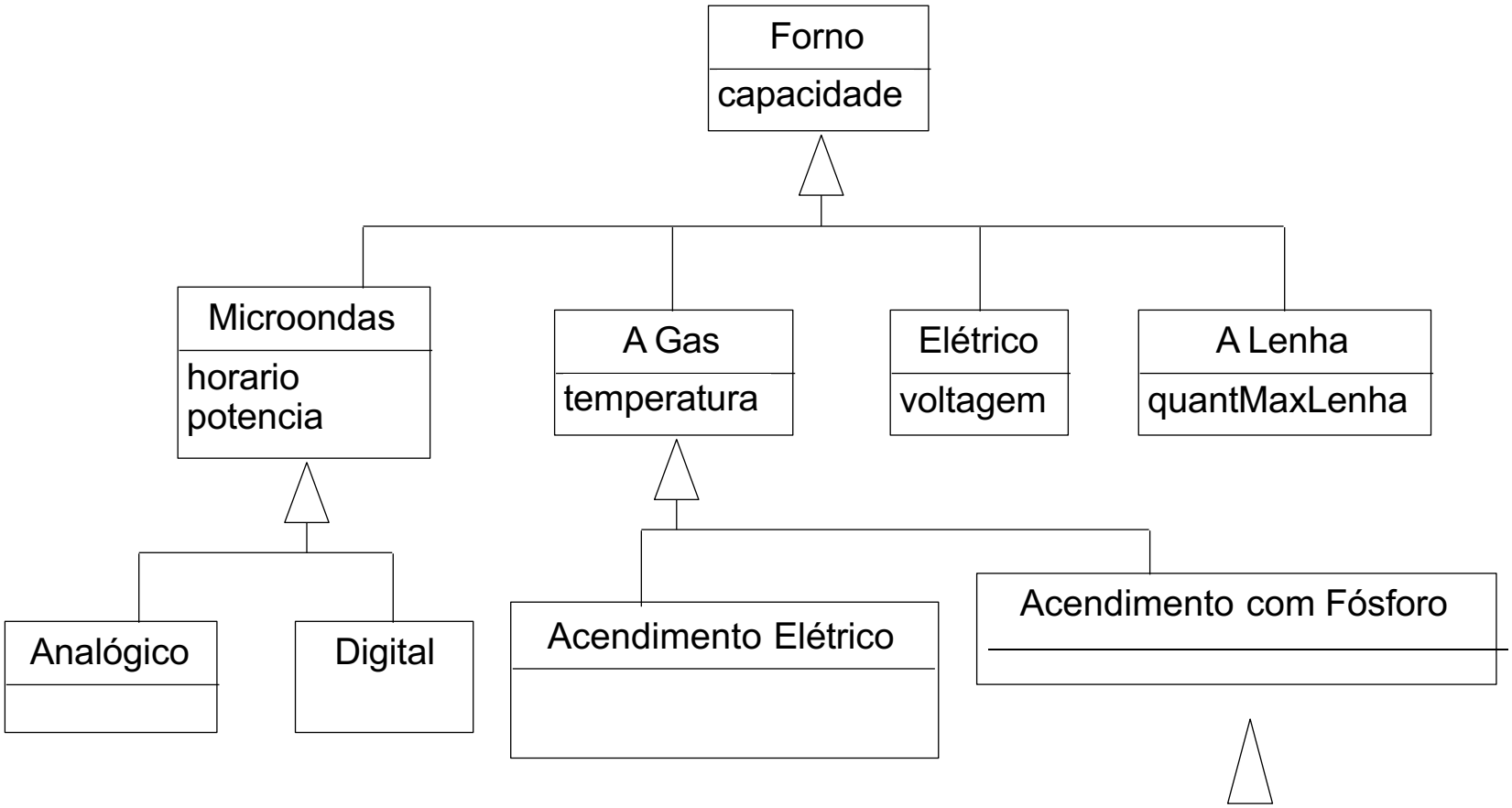
Herança

Na Orientação a Objetos as palavras classe base, supertipo, superclasse, classe pai e classe mãe são sinônimos, bem como as palavras classe derivada, subtipo, subclasse e classe filha também são sinônimos.

Herança: Generalização/Especialização



Herança


Forno
capacidade
Microondas
horario
potencia
A Gas
temperatura
Elétrico
voltagem
A Lenha
quantMaxLenha
Analógico
Digital
Acendimento Elétrico
Acendimento com Fósforo

Polimorfismo

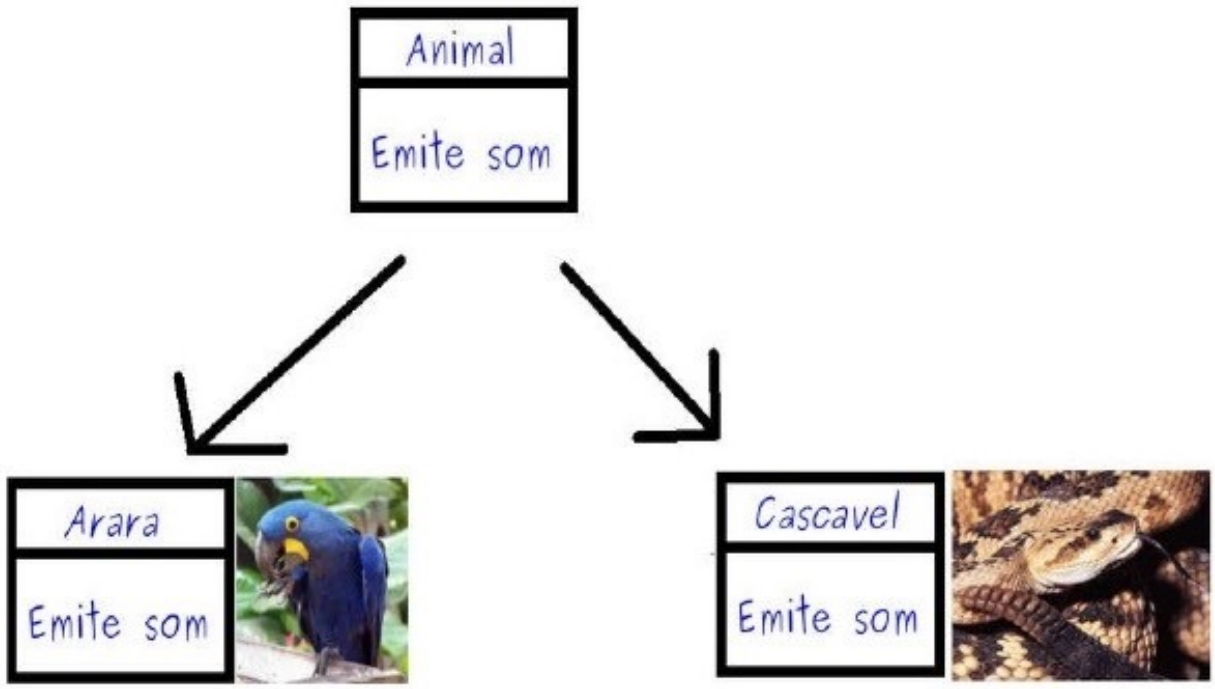
Formalmente polimorfismo quer dizer “várias formas”.

No caso da OO, polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem, dependendo do seu tipo de criação.

O poliformismo é alcançado com auxílio do uso de herança nas classes e a reescrita (*overriding*) de métodos das superclasses nas suas subclasses.

Duas subclasses de uma mesma classe podem ter implementações completamente diferentes de um mesmo método, o que leva os objetos a se comportarem de forma diferente, dependendo do seu tipo (classe).

Polimorfismo



Polimorfismo

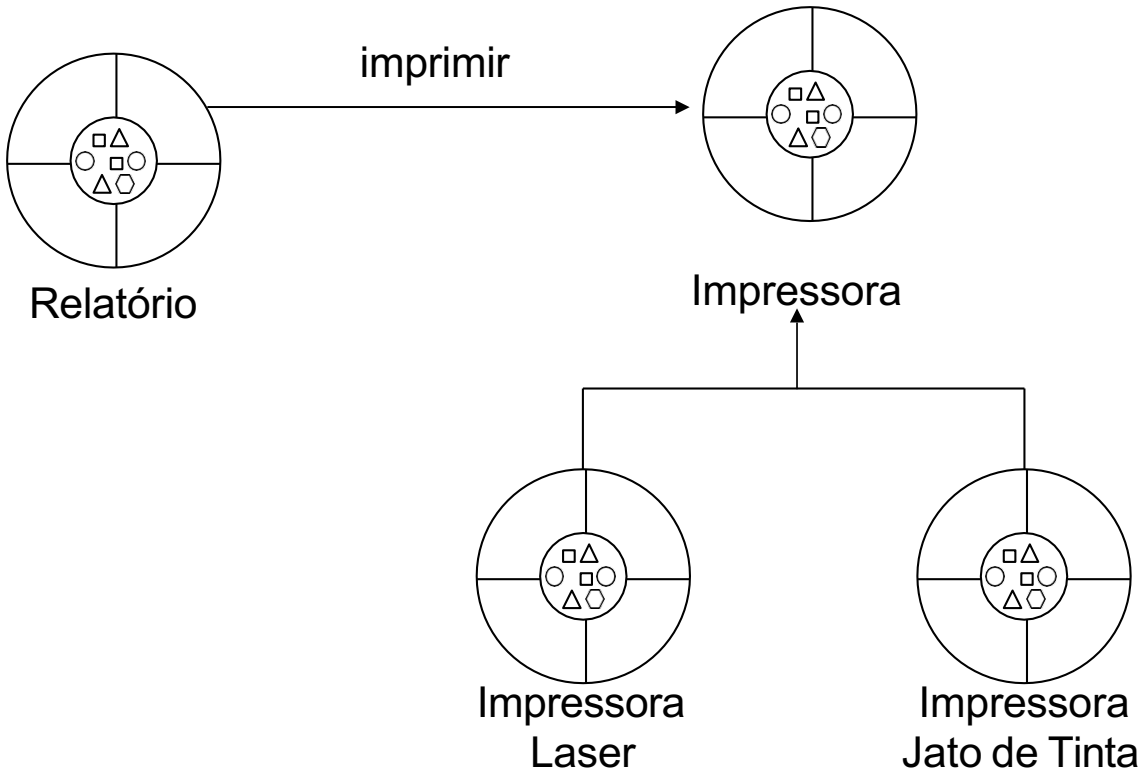
Exemplificando:

Podemos imaginar um programa que faça a impressão de um relatório, por meio de uma classe chamada Impressora, que é uma interface de acesso às funcionalidades da impressora usada, por meio de um driver fornecido pelo fabricante.

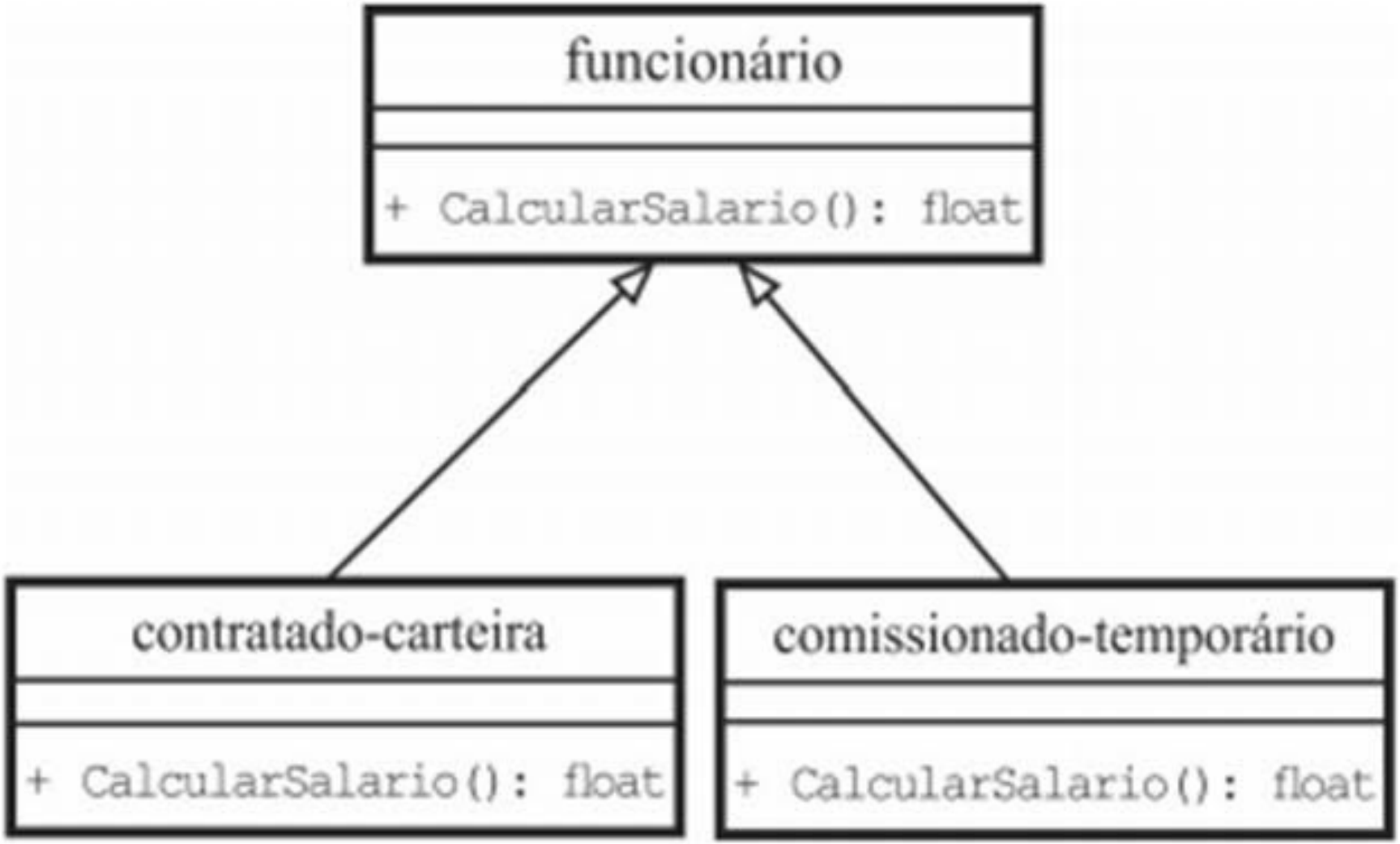
Uma impressora a laser tem um mecanismo de impressão totalmente diferente de uma impressora a jato de tinta, mas isso não importa para o programa.

Ele manda uma simples mensagem de imprimir para a impressora, e o modo como a impressora imprime no papel varia de acordo com o tipo de impressora usada, ou seja, a impressão se dá de formas diferentes para a mesma mensagem de imprimir.

Polimorfismo



Polimorfismo



Encapsulamento

Na OO, **encapsulamento** é o mecanismo utilizado para disponibilizar métodos que operam sobre os dados e que protegem o acesso direto indevido aos atributos de uma instância fora da classe onde estes foram declarados.

Esta proteção consiste em se usar **modificadores de acesso** mais restritivos sobre os atributos definidos na classe e fornecendo métodos que alteram os valores destes atributos de alguma forma.

O encapsulamento ajuda a prevenir o problema de interferência externa indevida sobre os dados de um objeto, como objetos que possam alterar os dados de outros objetos indevidamente.

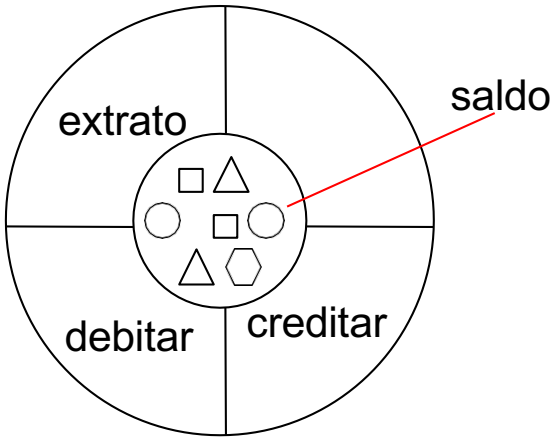
Encapsulamento

Um exemplo deste problema pode ser o saldo da conta bancária.

O saldo certamente não pode ser alterado ou manipulado diretamente, mas sim através de métodos adequados para isso, como métodos que fazem lançamentos de débitos e créditos.

A alteração direta do saldo causaria um problema de cálculos e inconsistência de dados.

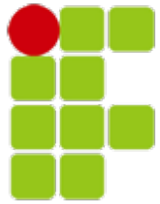
Justamente por isso devemos criar classes bem encapsuladas, que fornencem métodos adequados para operar sobre os dados dos objetos daquela classe.



Conta Corrente

O uso de encapsulamento também evita que um programa torne-se tão interdependente que uma pequena mudança tenha grandes efeitos colaterais.

Visibilidade – Modificador de acesso



- Private

  Somente a classe tem acesso

  Não é transmitido por herança

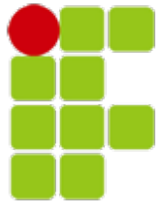
~ Default ou Friendly

  Acesso a classe inteira

    Visível para as classes do mesmo pacote

  Só é transmitido por herança em classes do mesmo pacote

Visibilidade



Protected

Visível em toda a classe

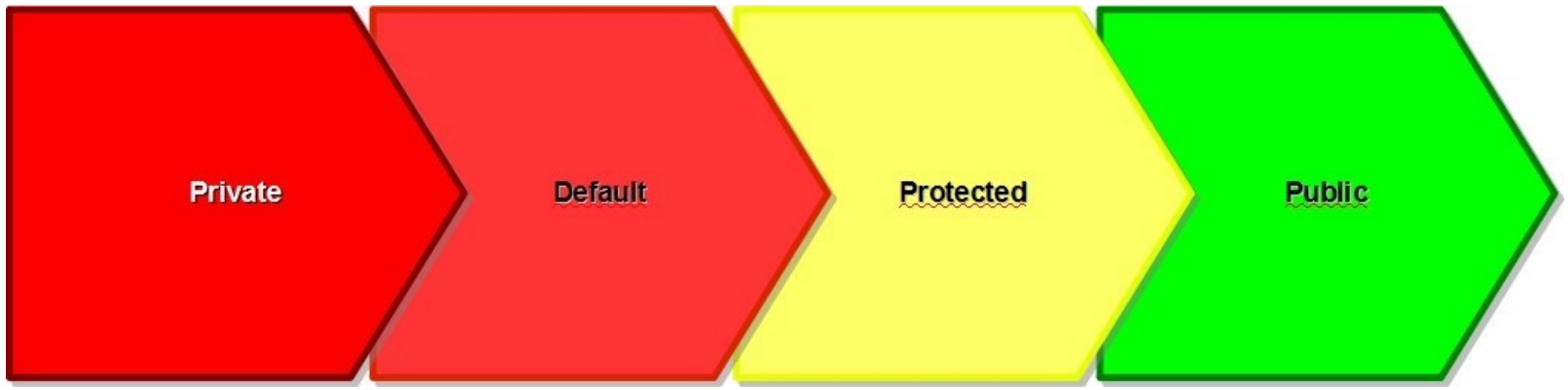
Visível em todas as classes de um pacote

Transmitido por herança

+ Public

Visível irrestritamente

Visibilidade



Conclusão

O paradígma da Orientação a Objetos traz um ganho significativo na qualidade da produção de software, porém grandes benefícios são alcançados quando as técnicas de programação OO são colocadas em prática com o uso de uma tecnologia que nos permita usar todas as características da OO; além de agregar à programação o uso de boas práticas de programação e padrões de projeto (*design patterns*).

Dúvidas?

Atividade

1 - Identifique as classes, atributos e métodos necessários para modelar e implementar:

a) Uma conta corrente que possui um número, um saldo, um status que informa se ela é especial ou não, um limite e um conjunto de movimentações.

Classe: Conta

Atributos: numero, status, limite, movimentação

Método: obterSaldo

Atividade

2 - Na orientação a objetos, o conceito que garante que nenhum acesso direto é concedido aos dados é atribuído por meio do(a): Justifique!!!

A. polimorfismo.

B. herança.

C. agregação.

D. abstração.

E. encapsulamento.

Atividade

3 - Considere as seguintes informações:

- carro (objeto)
- tamanho
- capacidade porta-malas
- quantidade de portas
- trava porta
- destrava porta

Nas informações listadas acima, referentes ao objeto carro, a informação trava porta, quando analisada em relação à orientação a objetos, representa: **Justifique!!!**

A. uma classe

B. um método

C. uma herança

D. uma estrutura

E. um atributo

Atividade

5 - Na programação orientada a objetos, a instanciação de objetos tem o objetivo de: Justifique!!!

A. criar um objeto.

B. definir atributos e métodos de uma classe.

C. inicializar uma classe.

D. abstrair os atributos de um objeto.