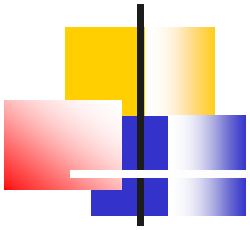
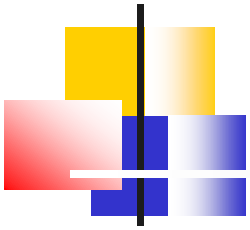


INF 213 – Estrutura de Dados
Prof. Marcus V. A. Andrade

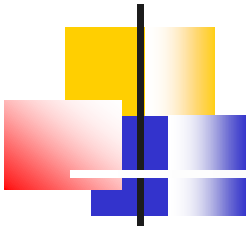
Introdução à Programação orientada a objetos (POO)



Programação Estruturada

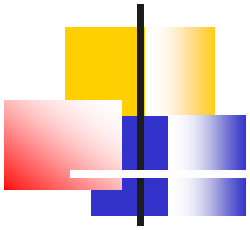
- Uma das primeiras técnicas de desenvolvimento de programas foi a programação estruturada ou procedimental;

- Um programa estruturado (ou procedimental) consiste de uma lista de instruções criadas pelo programador que deve ser executada pelo computador;



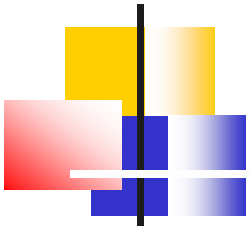
Programação Estruturada

- Para facilitar a organização do programa, as instruções são agrupadas (divididas) em funções;
- Esta estruturação (agrupamento das instruções) tem o objetivo de 'modularizar' os programas.
- Cada módulo consiste de um grupo de funções;
- Exemplos de linguagens procedimentais:
  - Fortran, Pascal e C



Programação Estruturada

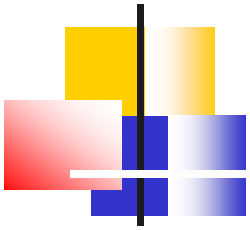
- Algumas vantagens da programação estruturada:
  - Tornar os programas mais compreensíveis;
  - Quaisquer correções de erros ou melhorias realizadas nas funções se refletem em todo o software;
  - Possibilitam maior organização de código e, portanto, facilitam a documentação.



Problemas na Programação Estruturada

- A programação estruturada coloca ênfase nas funções como:
  - Ler dados, processar, checar existência de erros, mostrar resultados, ...

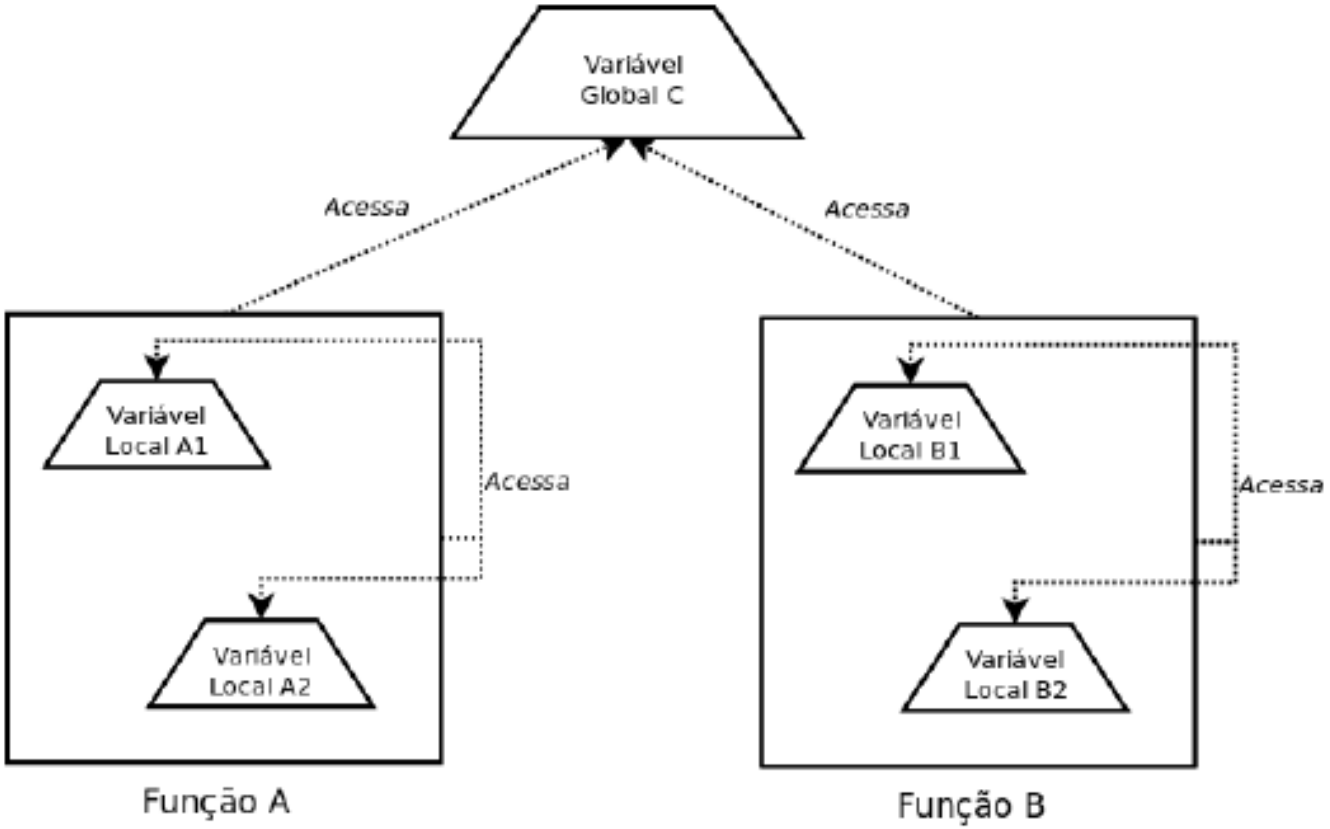
- Parte importante na solução de problemas computacionais, os DADOS, são relevados a um segundo plano;



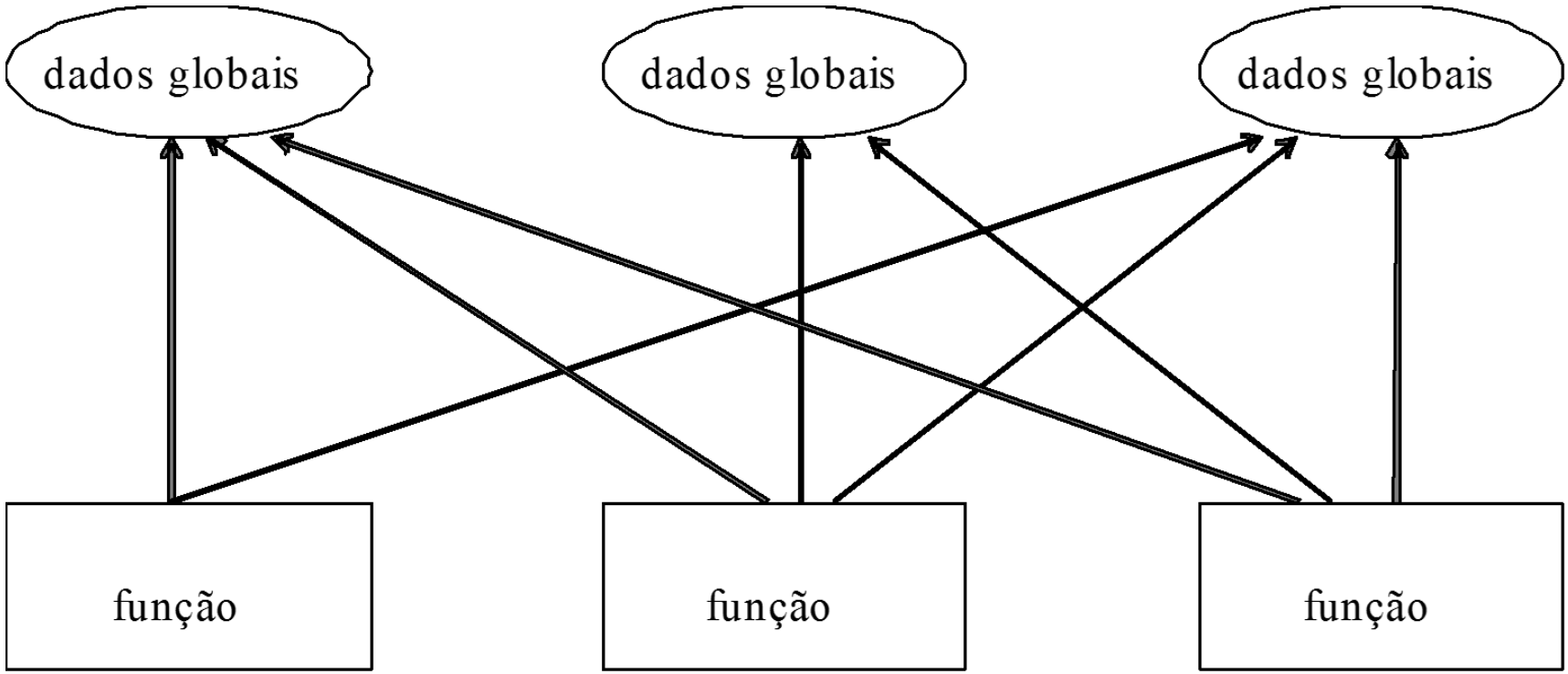
Problemas na Programação Estruturada

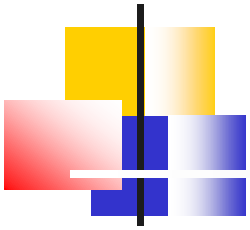
- Em linguagens procedimentais:
  - Variáveis locais são usadas apenas dentro do escopo da função onde são declaradas.
  - Variáveis locais não são úteis para dados que devem ser acessados por muitas e diferentes funções.
- Em outro contexto
  - Se muitas funções tem acesso a um conjunto de dados, então qualquer modificação nos dados requer modificações em todas as funções.
  - Torna-se difícil fazer a manutenção correta de tais funções.

Variáveis globais x variáveis locais



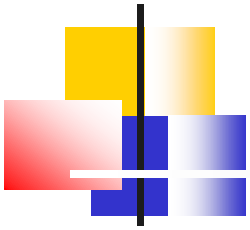
Funções e dados





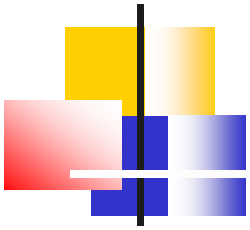
Problemas na Programação Estruturada

- Programas em linguagens procedimentais são mais difíceis de serem projetados
  - Funções não modelam adequadamente o mundo real;
- Mais difícil oferecer extensibilidade a linguagem procedimental, i.e.:
  - Adicionar, manipular e fazer a manutenção de novos tipos de dados.



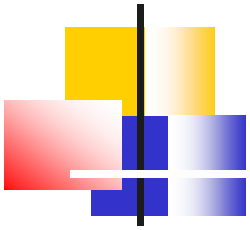
Introdução a Orientação a Objetos

- A Programação Orientada a Objetos (POO) foi desenvolvida para tentar contornar as limitações da programação estruturada;
- Para programas pequenos, não há necessidade de princípio organizacional;
- À medida que cresce o tamanho dos programas, torna-se mais difícil compreender os programas;



Introdução a Orientação a Objetos

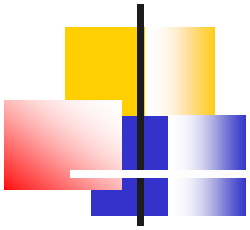
- A idéia básica da Programação Orientada a Objetos é restringir o acesso aos dados;
- Esconder os dados de todas funções, exceto daquelas (poucas) funções "críticas";
- Benefícios resultantes:
  - Proteção aos dados
  - Simplificação da manutenção



Introdução a Orientação a Objetos

- Assim, a POO define uma nova abordagem de conceber e construir programas;

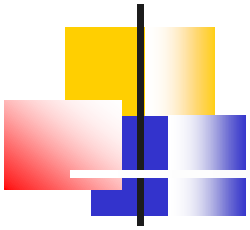
- As principais propostas:
  - permitir lidar com a complexidade;
  - facilitar a manutenção;
  - possibilitar a obtenção de programas mais confiáveis;



Introdução a Orientação a Objetos

- Linguagens OO combinam numa única entidade (objeto) dados e funções que atuam sobre esses dados.

- Funções de um objeto oferecem a única forma de acesso a seus dados:
  - Em C++ são chamadas de funções membro
  - Em Java são chamadas de métodos

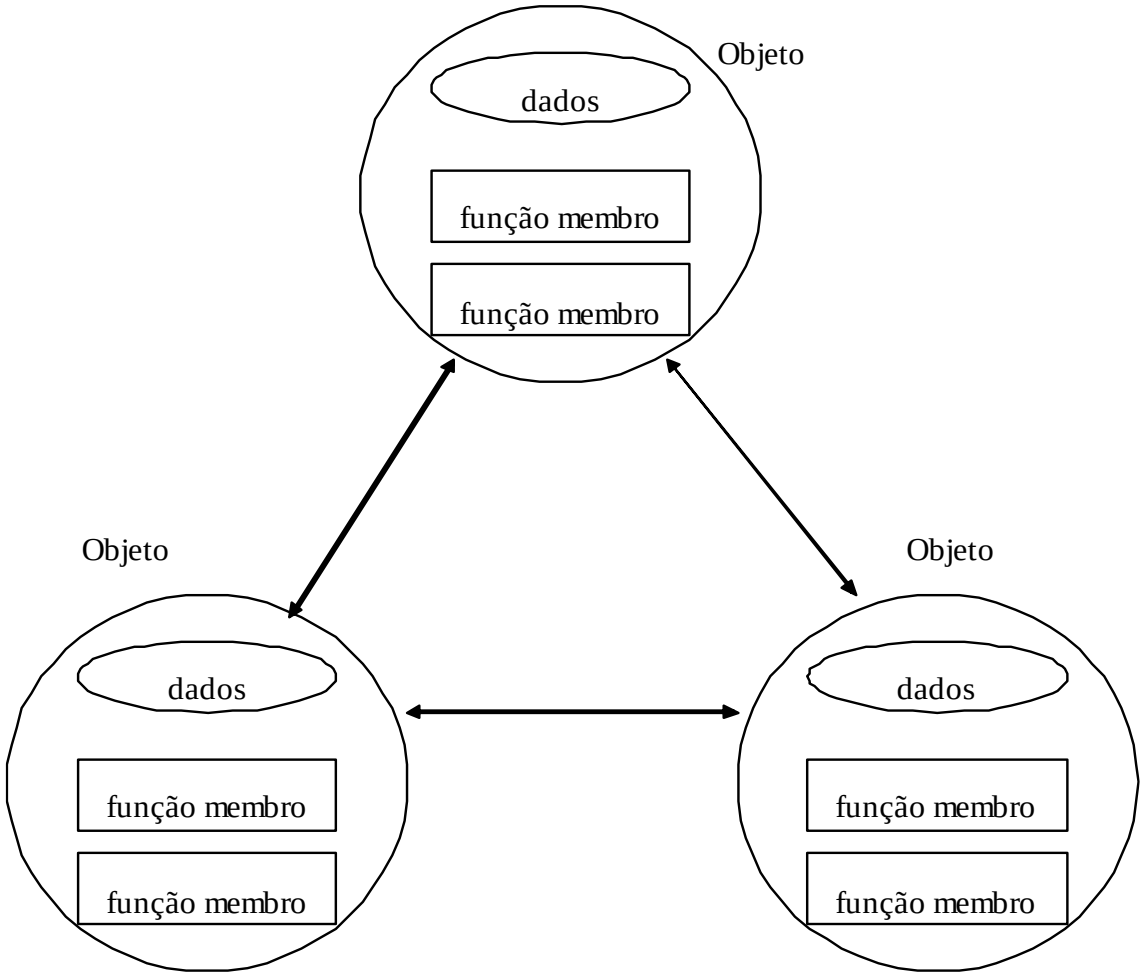


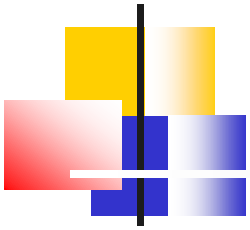
Introdução a Orientação a Objetos

- Para acesar os dados de um objeto, é necessário saber quais métodos/funções interagem com ele.
  - Isto simplifica escrita, depuração e manutenção de programas.

- Um programa numa linguagem OO consiste de vários objetos, comunicando-se entre si :
  - Através da chamada de funções membro de outro objeto (C++)
  - Através de trocas de mensagens (Java)

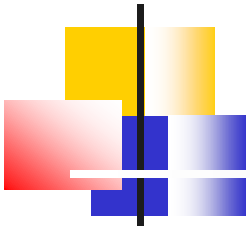
Introdução a Orientação a Objetos





Introdução a Orientação a Objetos

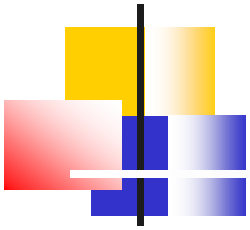
- A modelagem deve ocorrer em termos de objetos;
- Exemplos de objetos
  - Automóveis numa simulação de fluxo de tráfego
  - Funcionários num sistema de gestão de recursos humanos
  - Alunos e professores num sistema acadêmico



Classes

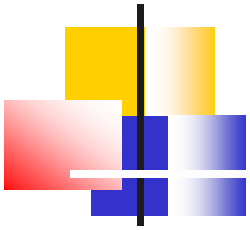
- O Projeto e a Programação Orientada a Objetos se baseia no conceito de classes;

- Objetos pertencentes a uma mesma classe (informalmente, “um mesmo tipo") possuem as mesmas características (atributos + comportamentos), dentro do contexto que se esta trabalhando.



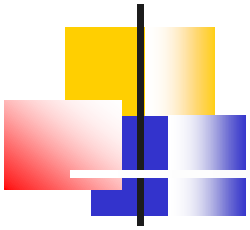
Classes

- Por exemplo:
  - Em uma noção geral, carros, ônibus, motocicletas, bicicletas (objetos) são todos veículos (classes).
  - O que caracteriza um veículo no software?
  - Um patins é um veículo?
- Assim, uma classe é uma abstração que descreve um grupo de objetos com as mesmas características.



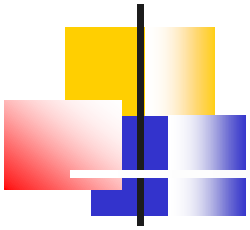
Abstração

- Mecanismo utilizado na análise do problema;
- Deve capturar as informações essenciais das entidades, elementos ou objetos envolvidos no contexto do sistema que esta sendo desenvolvido;
- Quais informações são relevantes vão depender da finalidade do software;
- Foco: visão geral e externa do objeto, separando seu comportamento de sua implementação.



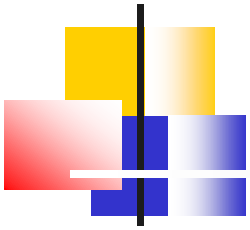
Abstração

- Exemplo:
  - Em um sistema de compras on-line, o usuário deve ser representado por seu nome, CPF, data de nascimento, sexo, e-mail, endereço e telefone.
  - Outras informações, que também poderiam representar este usuários como por exemplo, cor dos olhos, peso, número do título de eleitor etc. são irrelevantes para este sistema;



Abstração

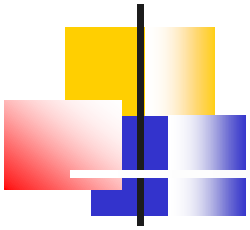
- Na etapa de projeto não deve ser levado em conta como estes dados serão representados;
- Deve-se identificar apenas quais dados são essenciais para representar o usuário e quais operações que o usuário pode fazer ou que podem ser feitas sobre a entidade usuário naquele sistema;
- Outras operações podem até estar relacionadas aos usuários, mas não representam seu papel neste sistema;



Classes x Objetos

- Classes descrevem os objetos definindo quais dados e funções pertencerão aquela classe;

- Definir uma classe não cria um objeto, assim como a simples existência de um tipo de dado não cria quaisquer variável;



Classes x Objetos

- Cada objeto é dito ser uma instância de uma classe;
- Os dados (informações) de um objeto são representados em atributos;
- O comportamento de um objeto é representado pelo conjunto de operações que podem ser executadas pelo ou com o objeto;

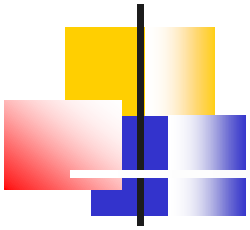
Classes x Objetos

- Uma classe descreve um conjunto (potencialmente infinito) de objetos individuais.

Classes x Objetos

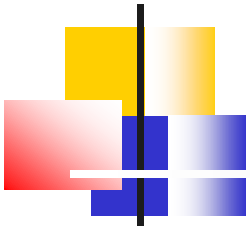
Classe Pessoa
nome idade altura peso

Objetos



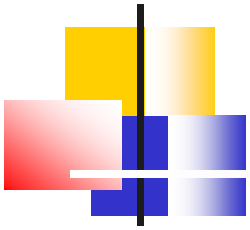
Classes x Objetos

- Cada objeto tem seus próprios valores para cada atributos;
- E compartilham entre si os nomes de seus atributos e suas operações;
- Em termos de programação, cada objeto contém uma referência para sua classe, seu "tipo", isto é, ele "sabe" o que ele é;
- Em C++, os atributos são denominados membros de dados e os métodos, funções membros;



Técnicas de POO

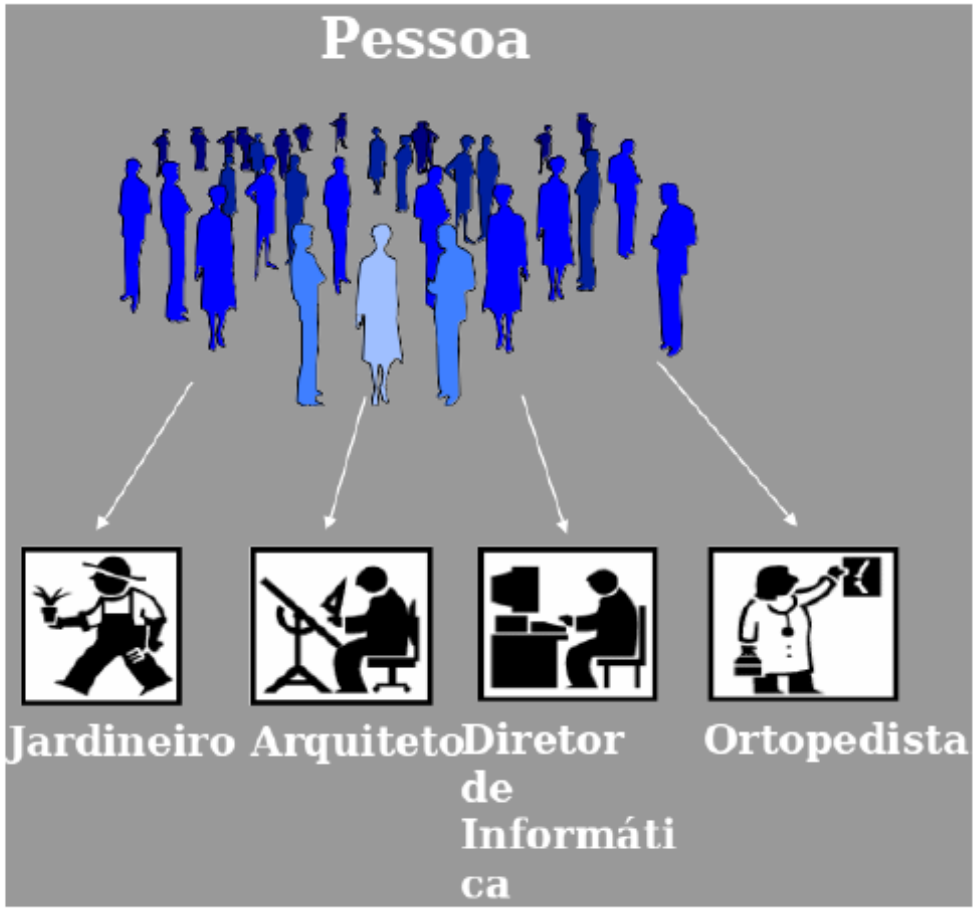
- Dentre as principais técnicas da Programação Orientada a Objetos temos:
  - Herança;
  - Polimorfismo;
  - Sobrecarga;
  - Classes genéricas



Herança

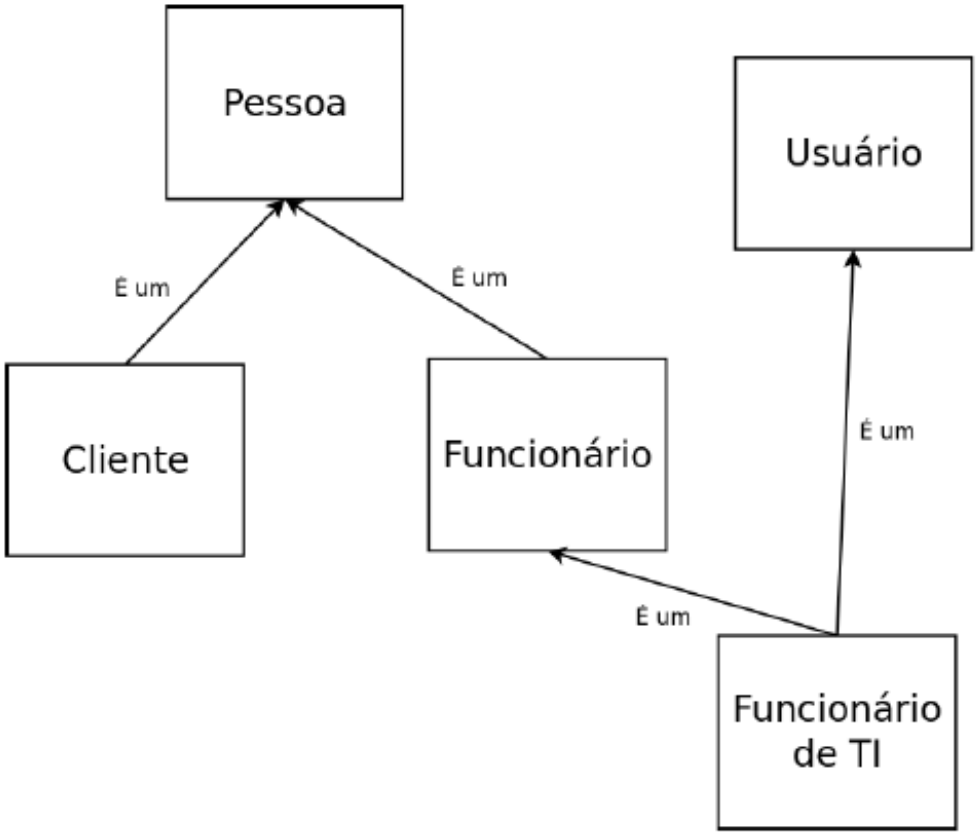
- Várias classes podem possuir características comuns e particularmente características específicas como, por exemplo:
  - As classes mamíferos, anfíbios, insetos, aves, ...
  - As classes carros de passeio, caminhões, ônibus, motocicletas, ...
- A ideia é identificar as características comuns das várias classes e criar uma classe *base* contendo estes atributos e métodos comuns;
- As características específicas são incluídas nas classes derivadas que compartilham as características comuns (da classe base)

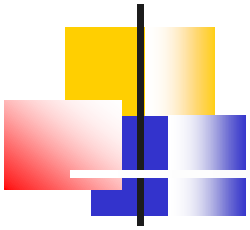
Herança simples



Herança múltipla

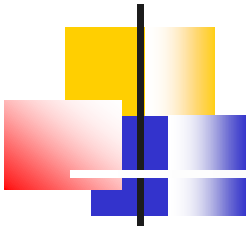
Exemplo do uso de herança





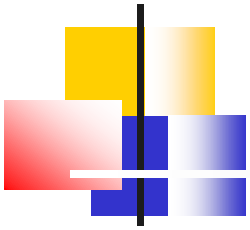
Polimorfismo

- Polimorfismo significa que uma mesma operação ou comportamento pode se manifestar de formas diferentes em situações diferentes;
- Isto é, um mesmo comportamento pode ser implementado de maneiras diferentes dependendo da situação;
- Por exemplo, uma função que determina a área de objetos geométricos: retângulo, triângulo, círculo, etc;



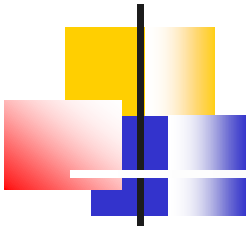
Sobrecarga

- Em varias linguagens é comum encontrarmos rotinas que fazem basicamente o mesmo processamento, porém, possuem nomes distintos porque se diferenciam apenas pelos tipos dos parâmetros.
- Exemplo: as funções somaDouble, somaInt e somaString realizam, essencialmente, a mesma operação, que é somar o valor recebido como parâmetro ao atributo da instância.



Sobrecarga

- A sobrecarga de funções ou métodos (*overloading*) consiste em criar diversos métodos (funções membro) com o mesmo nome, diferenciando-os por suas listas de argumentos (parâmetros).
- Métodos (funções membro) são identificados pela sua assinatura (protótipo): nome do método (função) + lista de parâmetros.
- Métodos com mesmo nome, mas com tipo, quantidade ou ordenação de parâmetros diferentes são considerados diferentes



Considerações gerais

- Os programas desenvolvidos seguindo o paradigma da orientação por objeto são:
  - Mais fáceis de serem mantidos;
  - Os vários módulos podem ser reutilizados;
  - Mais fáceis de serem testados

- Resumindo, a OO aumenta a produtividade no desenvolvimento de software