

Conceitos de Programação Orientada a Objetos

Por que a Orientação a Objetos?

- As abstrações podem corresponder às "coisas" do domínio do problema, facilitando o entendimento
- Esta abstração facilita a comunicação com os usuários
- Os mesmos objetos existem em todas as fases e uma notação única facilita a INTEGRAÇÃO ENTRE FASES de desenvolvimento
- A tecnologia de objetos facilita o entendimento do domínio do problema, permitindo o GERENCIAMENTO DA COMPLEXIDADE através da modularização
- Facilidade de mudanças através do ENCAPSULAMENTO de dados

Por que a Orientação a Objetos?

- ❑ Capacidade de aproveitar novas plataformas e ferramentas
- ❑ Facilidade de manutenção
- ❑ Economia de custos
- ❑ Encapsulamento das aplicações existentes
- ❑ Melhores interfaces
- ❑ Maior produtividade
- ❑ Participação no "futuro da computação"
- ❑ Prova da capacidade de usar a tecnologia
- ❑ Rápido desenvolvimento de aplicações estratégicas
- ❑ Reuso de software

Por que a Orientação a Objetos?

- Domínio do Problema (Mundo Real)

-
- Domínio da Solução (Software)

Orientação a Objetos - Conceitos

- ❑ OBJETO
 - ❑ MÉTODO
 - ❑ MENSAGEM
 - ❑ CLASSE
 - ❑ CLASSIFICAÇÃO
 - ❑ GENERALIZAÇÃO
 - ❑ ESPECIALIZAÇÃO
 - ❑ HERANÇA
 - ❑ POLIMORFISMO
 - ❑ SOBRECARGA
 - ❑ ENCAPSULAMENTO
 - ❑ ABSTRAÇÃO
 - ❑ MODULARIZAÇÃO
- PRECISO ENTENDER TODOS ESTES ITENS

OBJETO

- Entidades que possuem **dados e instruções** sobre como manipular estes dados
- Os objetos estão ligado à solução do problema.
 - Software Gráfico – Objetos: Círculos; Linhas; etc.
 - Software BD – Objetos: Tabelas; Linhas; Campos; etc.
 - Software Comercial: Pedidos; Produtos; Clientes; etc.
- Na OO a solução do problema consiste em um primeiro momento estabelecer quais os objetos serão necessários.
- Um objeto representa uma entidade: física, conceitual ou de software

OBJETO

- Os dados mantidos pelo objeto são chamados de atributos(propriedades)
- Os atributos de um objeto representam seu ESTADO, ou seja, o valor de seus atributos em um determinado momento.
- Objetos possuem IDENTIDADE, ou seja, cada objeto é diferente do outro e cada um tem seu próprio tempo de vida
- Cada objeto tem uma identidade única, mesmo que o estado seja idêntico para ambos os objetos

OBJETO

■ Dados ligados ao objeto – Exemplos:

- Círculo – ponto_centro, raio
- linha – ponto_inicio; ponto_final
- Cliente – Nome; Data Nascimento; Telefone
- Telefone – Numero; Modelo; Cor
- Exemplos:

//criação de objetos (sintaxe C++)

Ponto p(3,4);

Circle c(p,5.4);

Cliente pessoa;

tbl_cliente ==> OBJETO

cli_codigo ==> ATRIBUTO

cli_nome ==> ATRIBUTO

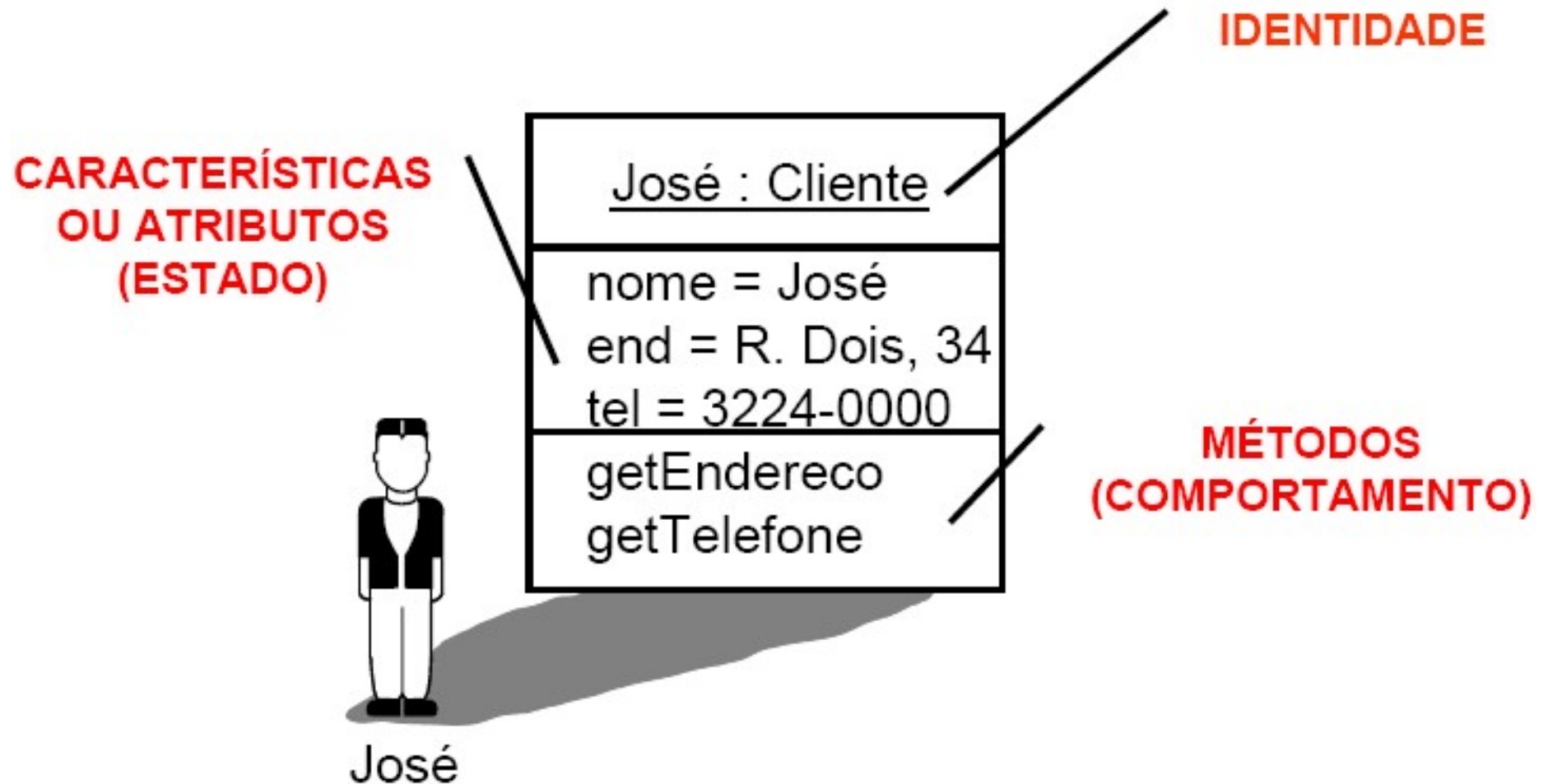
MÉTODOS

- ❑ Métodos são procedimentos que determinam como o objeto se comporta.
- ❑ Através dos métodos é possível manipular os dados contidos no objeto.
- ❑ Os métodos estão ligados ao comportamento do objeto
- ❑ Exemplos
 - Um círculo poderia possuir os métodos:
draw; move; getArea; getPerimeter; setCenter
 - Um cliente poderia possuir os métodos:
calculaldade; getTelefone
 - Um Telefone poderia possui os métodos:
tocar; disar

MÉTODOS

- ❑ Um método é a implementação de uma operação
- ❑ Métodos só tem acesso aos dados da classe para a qual foram definidos
- ❑ Métodos normalmente possuem argumentos, variáveis locais, valor de retorno, etc.
- ❑ Alguns métodos especiais:
 - Construtores – Criam objetos
 - Destrutores – Destroem objetos
 - Acessores – Recuperam o estado de um atributo (getNomeAtributo)
 - Modificadores – Alteram o estado de um atributo (setNomeAtributo)

OBJETOS - RESUMO



MENSAGEM

- ❑ Objetos se comunicam entre si através de mensagens.
- ❑ Uma mensagem é uma chamada de um método.
- ❑ A mensagem possui os seguintes componentes:
 - Receptor – nome do objeto que irá receber a mensagem
 - Método – Método do receptor que será utilizado
 - Argumentos – Informação adicional para a execução do método
 - **Exemplos**

Point p(0,0), pNewCenter(2,3);

Circle c(p,3);

c.getArea(); //Exemplo Mensagem

c.setCenter(pNewCenter); //Exemplo Mensagem

CLASSE

- ❑ Classe é um agrupamento de objetos
- ❑ A classe consiste nos métodos e nos dados que um determinado objeto irá possuir.
- ❑ Objetos são criados quando uma mensagem solicitando a criação é recebida pela sua classe.
- ❑ A programação orientada a objetos consiste em implementar as classes e na utilização das mesmas, através da sua intercomunicação.
- ❑ Um objeto é uma instância da classe.
- ❑ Os objetos de uma classe compartilham os mesmos atributos, operações, relacionamentos e semânticas
- ❑ Objetos só reagem a mensagens que fazem parte das ações do protocolo de sua classe

CLASSIFICAÇÃO

- Na POO classificação consiste em criar classes a partir dos objetos envolvidos em um determinado problema
- As classes podem ser criadas a partir do momento em que for possível isolar no domínio do problema objeto que possui atributos e métodos comuns
- Ex: Diferentes tipos de pessoas interagem com uma empresa

COMPORTAMENTO	CLASSE
Pessoas interessadas nos produtos	???
Pessoas que já compraram os produtos	???
Pessoas que são responsáveis por um grupo de trabalhadores	???
Pessoas responsáveis pela demonstração de produtos e sua venda	???
Pessoas que trabalham na linha de produção	???

Generalização e Especialização

■ GENERALIZAÇÃO

- A generalização consiste em obter similaridades entre as várias classes e partir destas similaridades, novas classes são definidas.
- Estas classes são chamadas **superclasses**

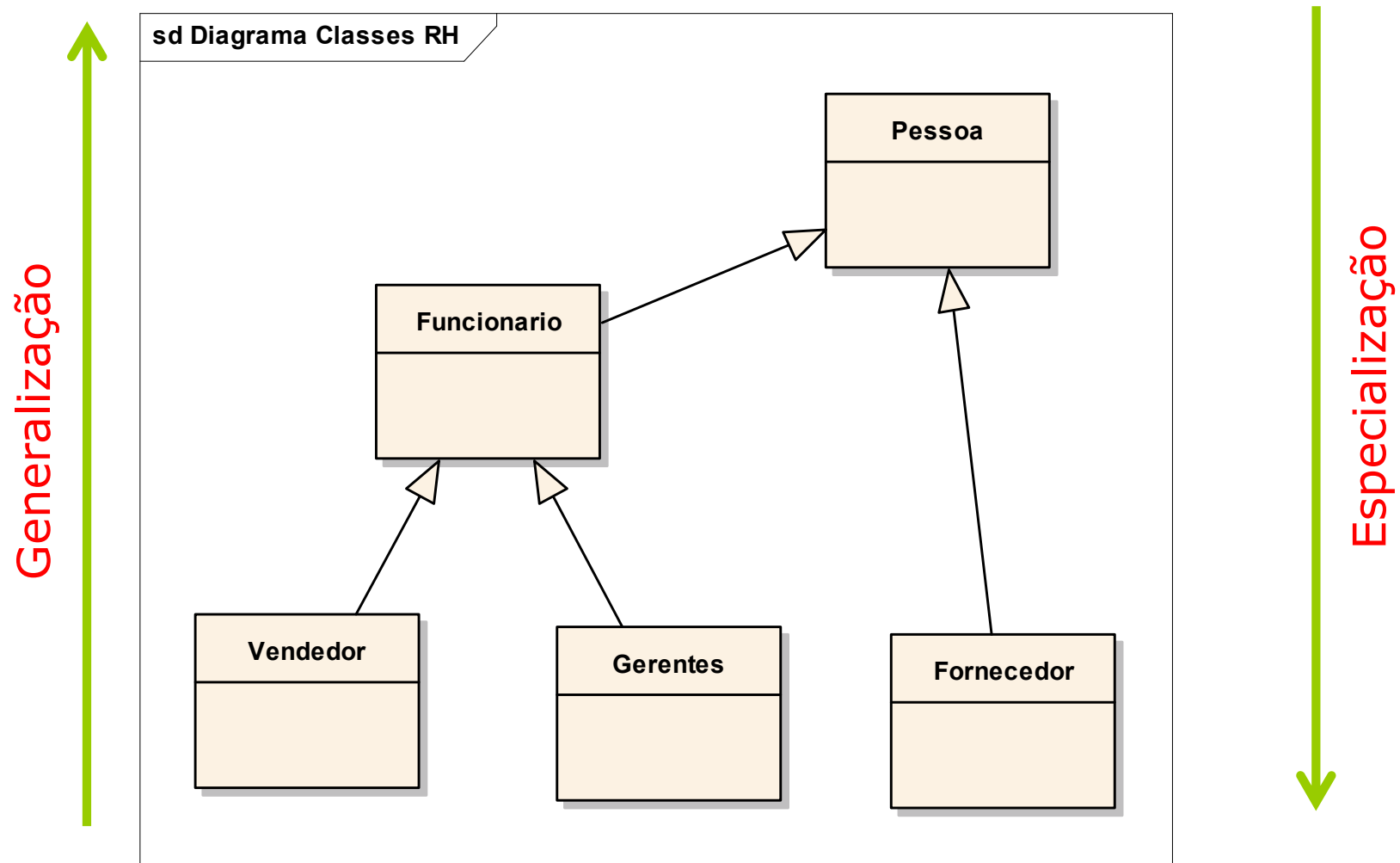
■ ESPECIALIZAÇÃO

- A especialização por sua vez consiste em observar diferenças entre os objetos de uma mesma classe e dessa forma novas classes são criadas.
- Estas classes são chamadas **subclasses**.

Generalização e Especialização

Exemplo

■ Hierarquia de classes



HERANÇA

- Herança é a capacidade de uma subclasse de ter acesso as propriedades da superclasse a ela relacionada.
- Dessa forma as propriedades de uma classe são propagadas de cima para baixo em um diagrama de classes.
- Neste caso dizemos que a subclasse herda as propriedades e métodos da superclasse
- A relação de herança entre duas classes é uma relação da seguinte forma: A "é um tipo de" B, onde A e B são classes.
- Caso esta relação entre as classes não puder ser construída, em geral, também não se tem uma relação de herança entre a classe A a partir da classe B.

HERANÇA

■ Exemplos:

- Um Carro de Passeio "é um tipo de " veículo; Um caminhão "é um tipo de" veículo;
- Um círculo "é um tipo de" uma figura geométrica; Um quadrado "é um tipo de" figura geométrica;
- Um vendedor "é um tipo de" Empregado; Um empregado "é um tipo de" pessoa.

■ Herança Múltipla

- Uma subclasse herda características de mais uma classe
- Exemplos
 - Um gerente de vendas "é um tipo" de vendedor e também "é um tipo de" gerente;

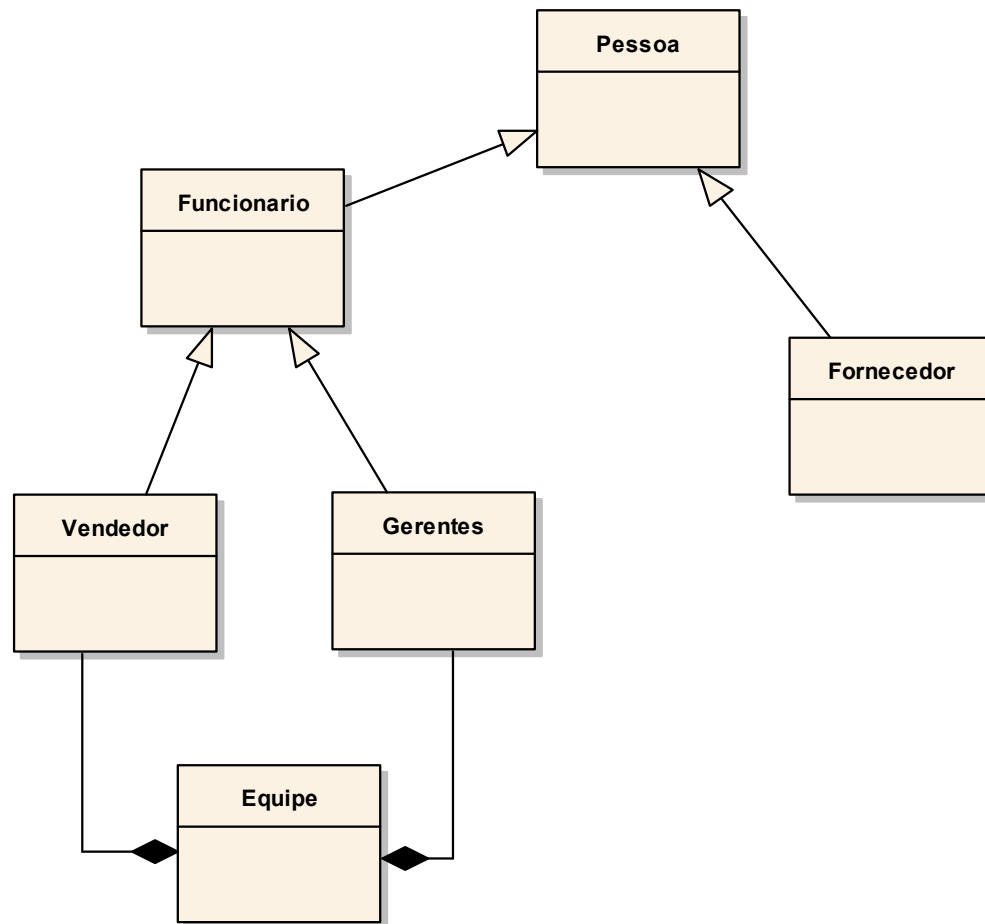
HERANÇA x USO

- Além da relação de herança entre as classes existe a relação de uso
- HERANÇA
 - classe A "é um tipo de" B
- USO / AGREGAÇÃO (Relação de Conteúdo)
 - classe D "contém" classe C"
 - classe D "usa" classe C"
 - classe C "é parte da" classe D
 - Exemplo: Uma **equipe contém um gerente e um grupo de vendedores**

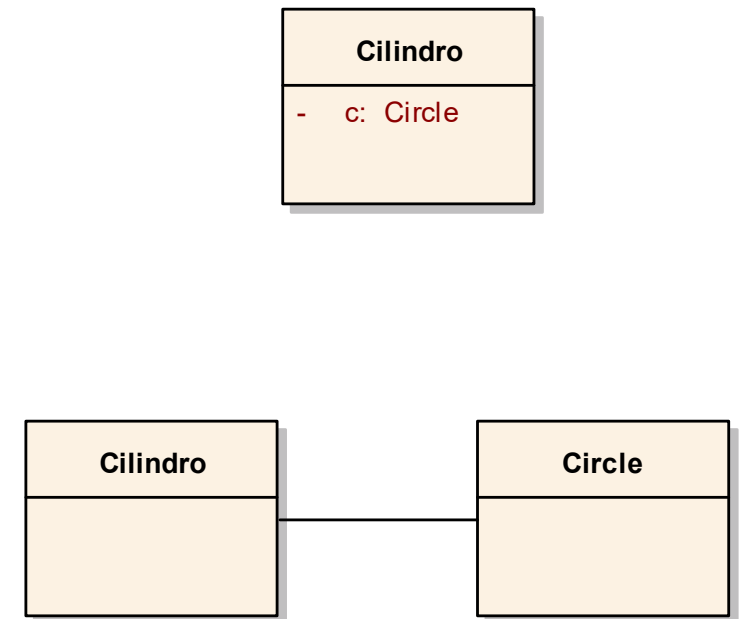
Herança x Uso

Exemplo

sd Diagrama Classes RH



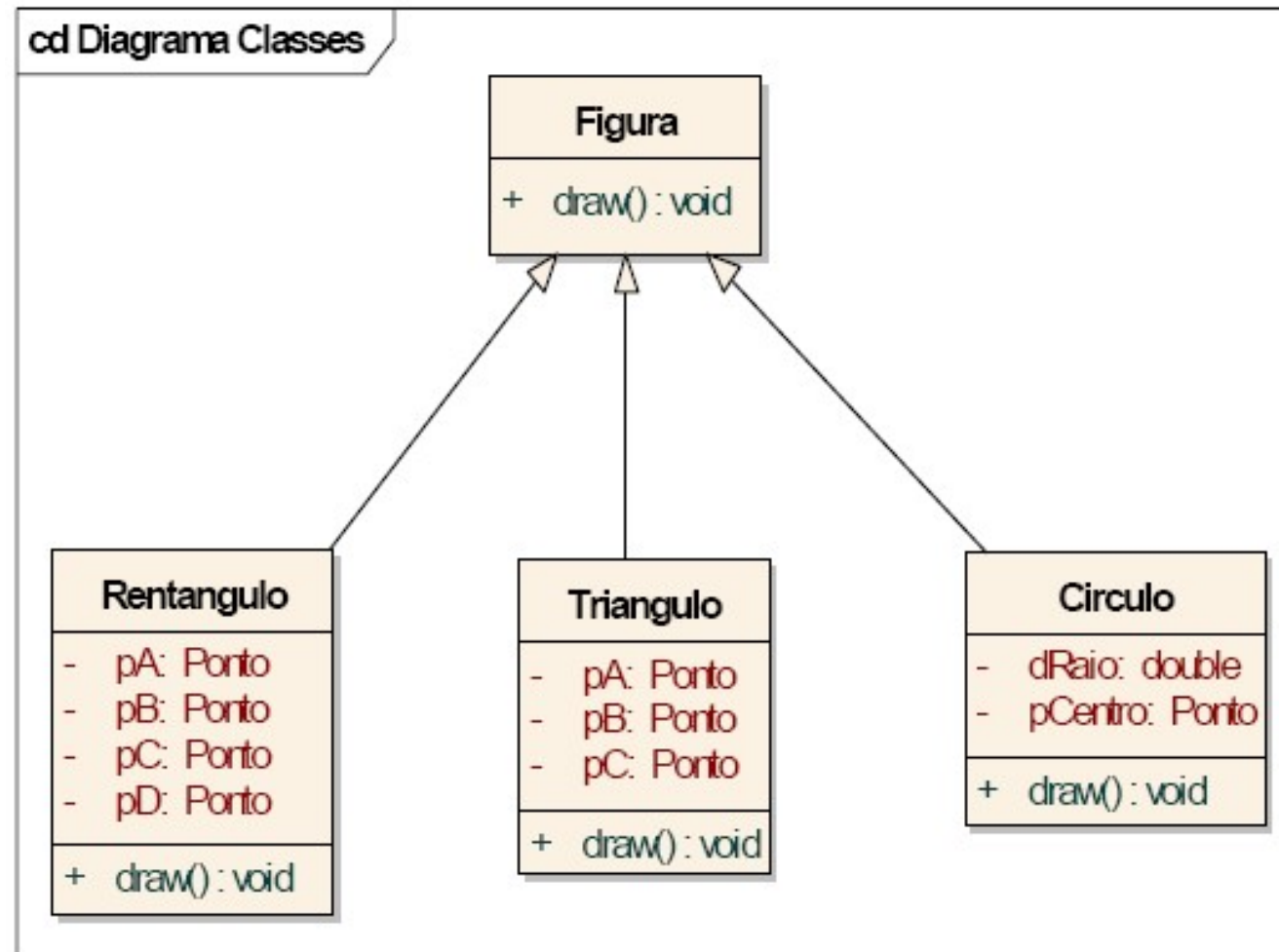
class Classes



POLIMORFISMO (Override)

- Os objetos respondem às mensagens que eles recebem através dos métodos.
- A mesma mensagem pode resultar em diferentes resultados. Esta propriedade é chamada de **polimorfismo**
 - **Exemplo: Método getSalario()**
 - Para um empregado qualquer → getsalario() = Salario;
 - Para o gerente → getsalario() = salario + bonificacao;
 - **Exemplo: Método draw()**
 - Para uma figura qualquer desenha uma forma não definida
 - Para o retângulo, triângulo e círculo o mesmo método responde de uma forma diferente

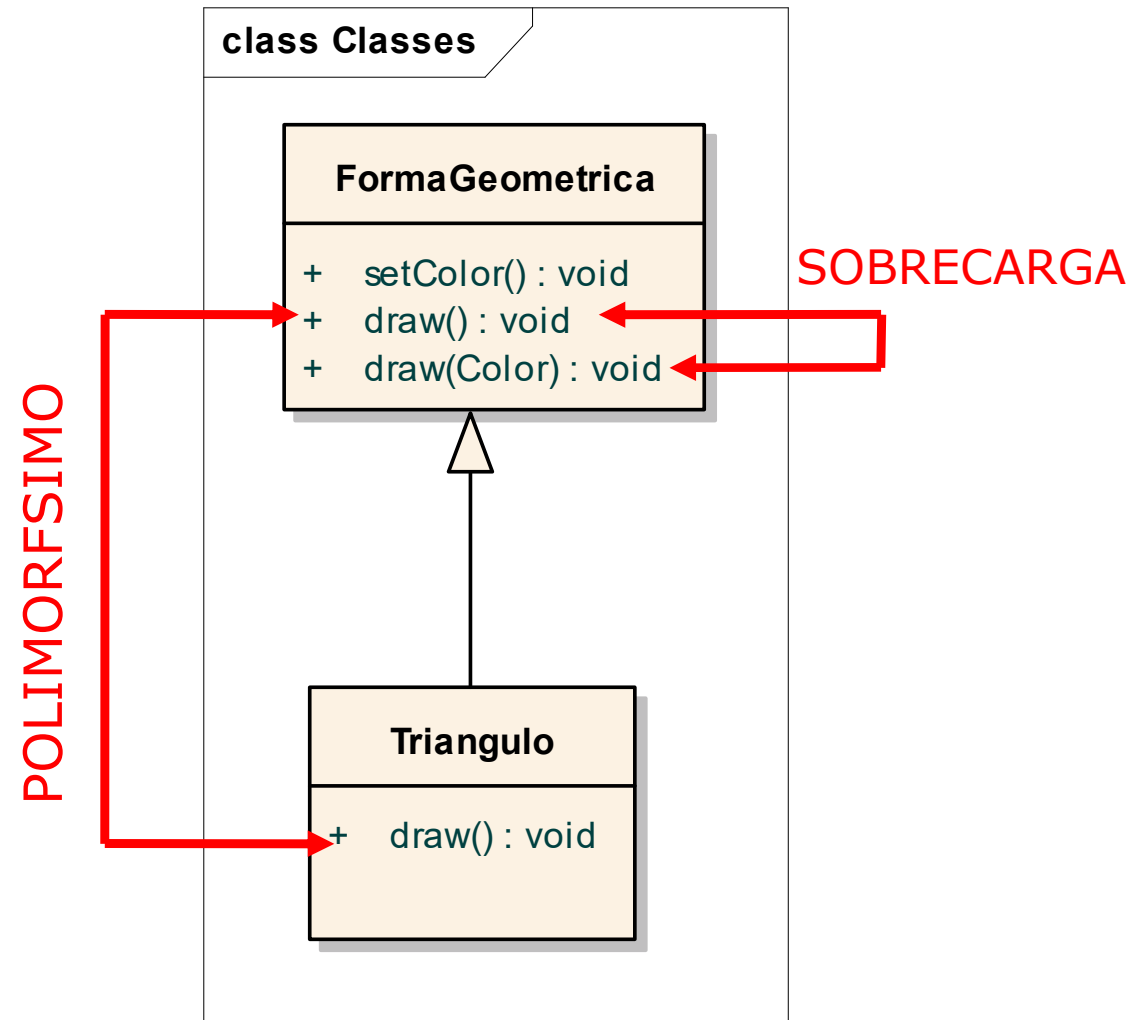
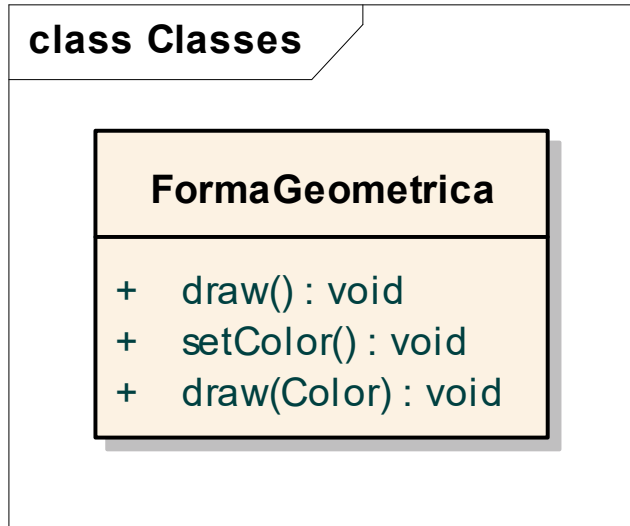
POLIMORFISMO - Exemplo



SOBRECARGA(Overload)

- Nas linguagens orientadas a objetos é possível, em uma classe, a existência de métodos que possuem o mesmo nome, porém com diferentes assinaturas.
- Este conceito é chamado de Sobrecarga (Overload)
 - **Exemplo: Em uma classe Figura, temos os seguintes métodos**
 - draw() → desenha o objeto e utiliza uma cor padrão
 - draw(Color) → desenha o objeto, porém recebe uma cor como parâmetro

SobreCarga - Exemplo



ENCAPSULAMENTO

- Conceito que indica que os dados contidos em um objeto somente poderão ser acessados e/ou modificados através de seus métodos.
- Dessa forma não é possível alterar os dados diretamente, somente através de métodos definidos no objeto
- Exemplo
 - O centro do círculo somente pode ser alterado/recuperado pelos métodos `setCenter`/`getCenter`.

ENCAPSULAMENTO

- O encapsulamento assegura que toda a comunicação com o objeto seja realizada por um conjunto pré-definido de operações
- O encapsulamento facilita as mudanças, visto que os objetos são isolados uns dos outros, reduzindo desta forma o acoplamento
- Além disso o encapsulamento facilita a manutenção de classes, bem como, garante a integridade dos atributos de um objeto em um determinado instante.

ABSTRAÇÃO

- Abstração é o processo de identificar as qualidades ou propriedades importantes do problema que está sendo modelado.
- Através de um modelo abstrato, pode-se concentrar nas características relevantes e ignorar as irrelevantes.
- Abstração é fruto do raciocínio.
- Através da abstração é possível controlar a complexidade. Isto é feito através da ênfase em características essenciais, fazendo-se uma supressão daquilo que não está ligado ao domínio do problema.

MODULARIZAÇÃO

- Consiste em decompor o problema em partes menores.
- Dessa forma o foco é mantido em itens(classes; pacotes; etc.) menores, coesos e fracamente acoplados.

Praticando os conceitos...

Atividade

- Utilizando os conceitos
 - Herança
 - Polimorfismo
 - Sobrecarga
 - Classe
 - Atributo
 - Método
- Propor uma modelagem relacionada ao seu ambiente de trabalho

Análise Orientada a Objetos (OOA)

Projeto Orientado a Objetos (OOD)

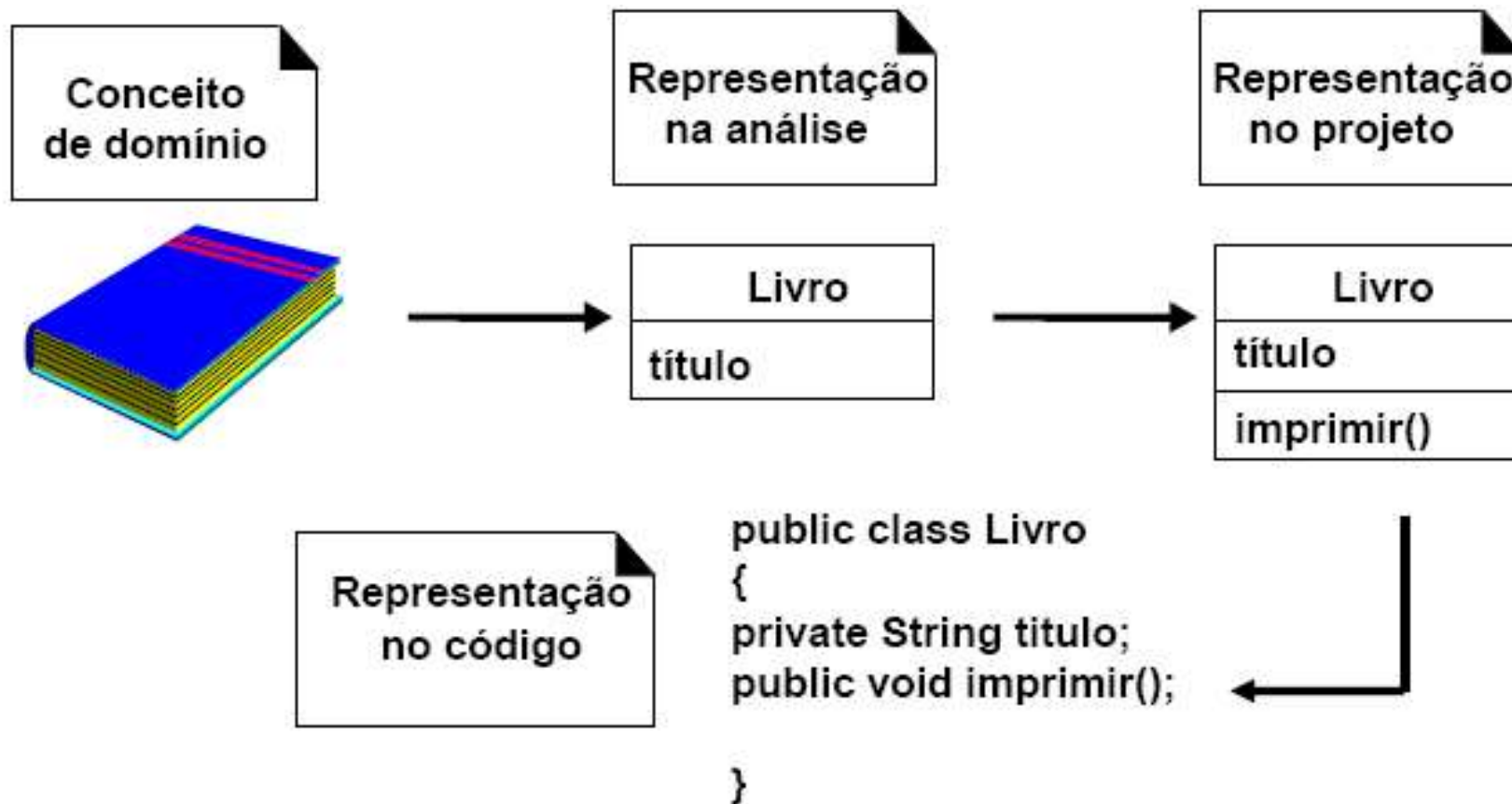
- As técnicas tradicionais (Análise e Projeto Estruturado) não são adequadas para o desenvolvimento de software utilizando a orientação à objetos.
- A utilização do orientação à Objetos solicita uma nova forma de abstrair e entender o problema.
- A linguagem UML é um padrão de diagramação para visualizar os resultados da análise e projeto orientados à objetos

Análise Orientada a Objetos (OOA)

Projeto Orientado a Objetos (OOD)

Exemplo

- O conceito "Livro" em um sistema de biblioteca



Análise Orientada a Objetos(OOA)

- Objetivo básico

- Identificar classes a partir das quais objetos serão representados como instâncias

- Envolve as seguintes tarefas

- Identificação de Objetos
 - Especificação de Atributos
 - Definição de métodos
 - Comunicações entre objetos

Análise Orientada a Objetos(OOA)

■ IDENTIFICAÇÃO DE OBJETOS

- Entidades externas (Outros sistemas; dispositivos;Pessoas)
- Coisas ligadas ao domínio do problema (Relatórios;Displays;...)
- Ocorrências ou Eventos (Conclusão de um movimento;Alarme disparado; Clique do mouse; etc.)
- Papéis ou funções (Engenheiro; Gerente; Vendedor) desempenhados por pessoas
- Unidades organizacionais (Grupo; Equipe;...)
- Lugares (Piso de fábrica; área de descarga)
- Estruturas (Sensores; veículos de quatro rodas;...)

Análise Orientada a Objetos(OOA)

■ IDENTIFICAÇÃO DE OBJETOS – CRITÉRIOS

1. RETENÇÃO DE INFORMAÇÃO – Objeto deve guardar informação que será utilizada pelo sistema
2. SERVIÇOS NECESSÁRIOS – Conjunto de operações identificáveis que podem mudar o valor de seus atributos
3. MÚLTIPLOS ATRIBUTOS – Objeto deve conter mais de um atributo
4. ATRIBUTOS COMUNS – Conjunto de atributos deve ser aplicado a todos os objetos
5. OPERAÇÕES COMUNS – Conjunto de operações devem ser aplicáveis a todos os objetos.
6. REQUISITOS ESSENCIAIS – Entidades externas que aparecem no espaço problema que consomem e/ou produzem informação

Análise Orientada a Objetos(OOA)

- Em uma especificação:
 - NOMES são potenciais objetos (e classes)
 - VERBOS são potenciais métodos
- A regra acima deve ser utilizada apenas como referência.
- O entendimento do contexto, das necessidades do usuário são fundamentais para classificar possíveis objetos e métodos