Orientação a Objetos
Revisão Conceitos

- CLASSE
- CLASSIFICAÇÃO
- GENERALIZAÇÃO
- ESPECIALIZAÇÃO
- HERANÇA
- INTERFACES
- POLIMORFISMO
- SOBRECARGA
- ENCAPSULAMENTO
- ABSTRAÇÃO
- MODULARIZAÇÃO

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

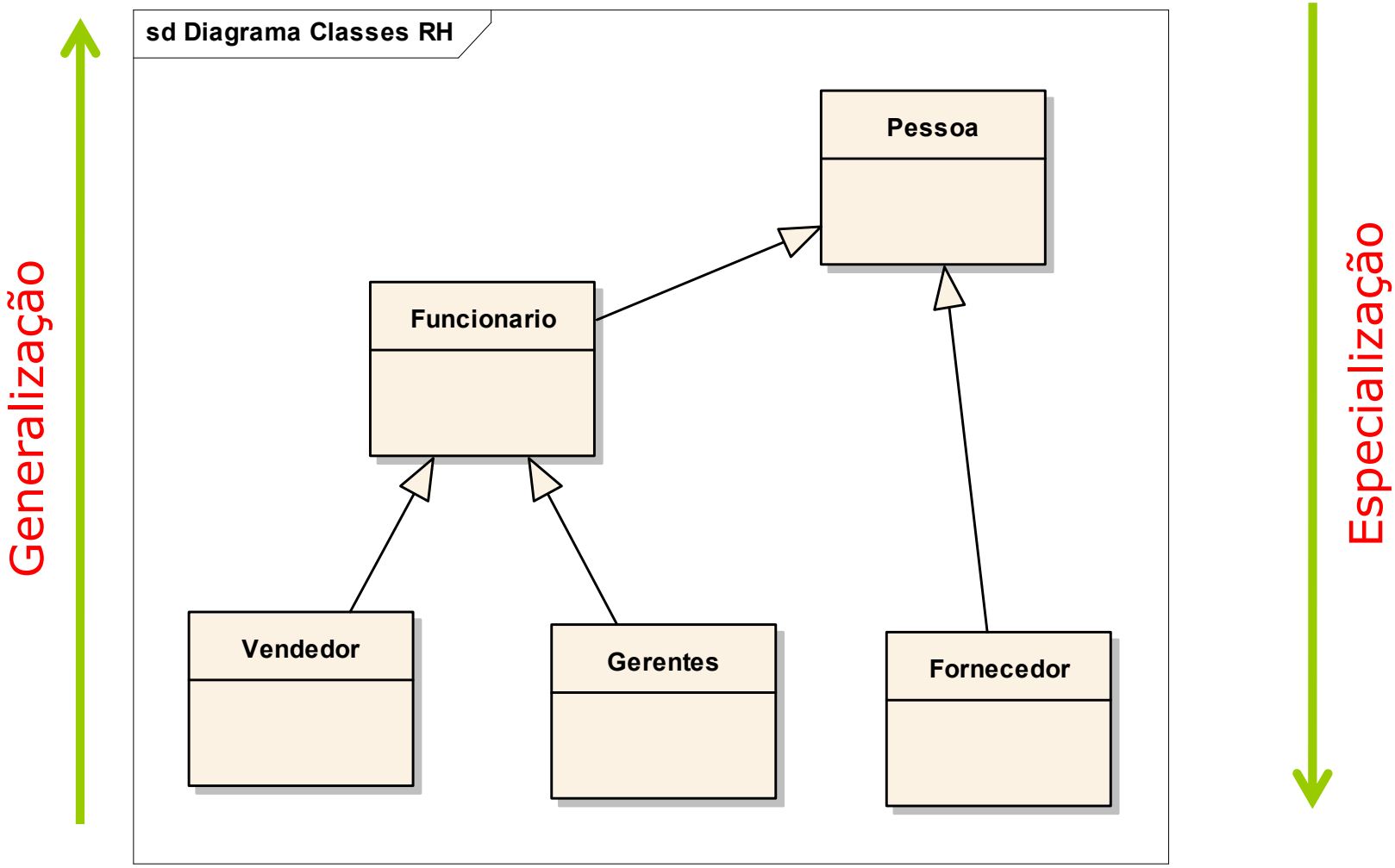
COMPORTAMENTO	CLASSE
Pessoas interessadas nos produtos	???
Pessoas que já compraram os produtos	???
Pessoas que são responsáveis por um grupo de trabalhadores	???
Pessoas responsáveis pela demonstração de produtos e sua venda	???
Trabalhadores da linha de produção	???

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 “e 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

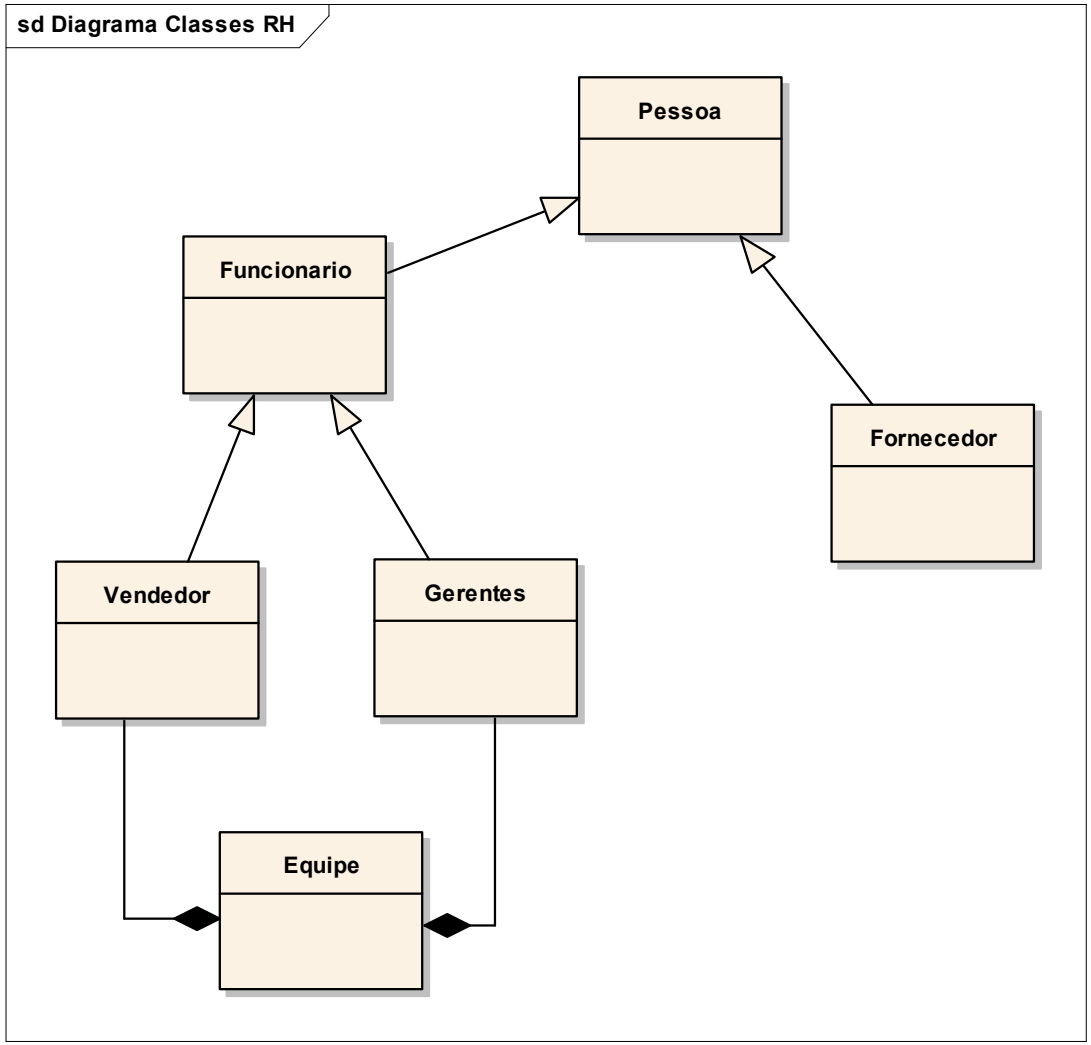
- Exemplos:
  - Um Carro de Passeio “é um tipo de ” veículo; Um caminhão “é um tipo de” veículo;
  - Um círculo “é um tipo de” uma figura geométrica; Um quadrado “é um tipo de” figura geométrica;
  - Um vendedor “é um tipo de” Empregado; Um empregado “é um tipo de” pessoa.
- Herança Múltipla
  - Uma subclasse herda características de mais uma classe
  - Exemplos
    - Um gerente de vendas “é um tipo” de vendedor e também “é um tipo de” gerente;

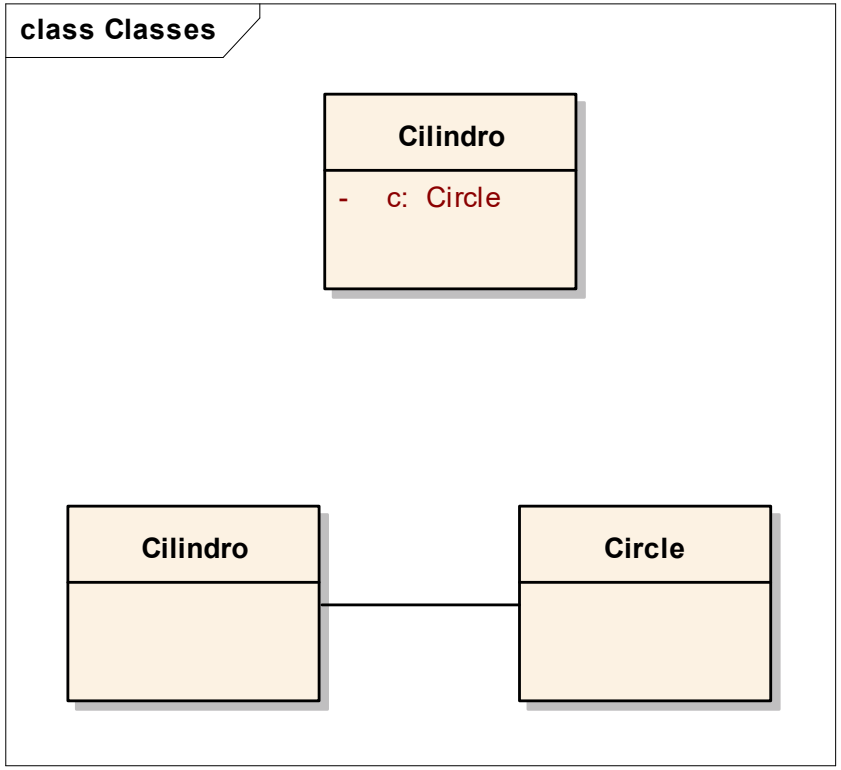
HERANÇA x USO

- 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

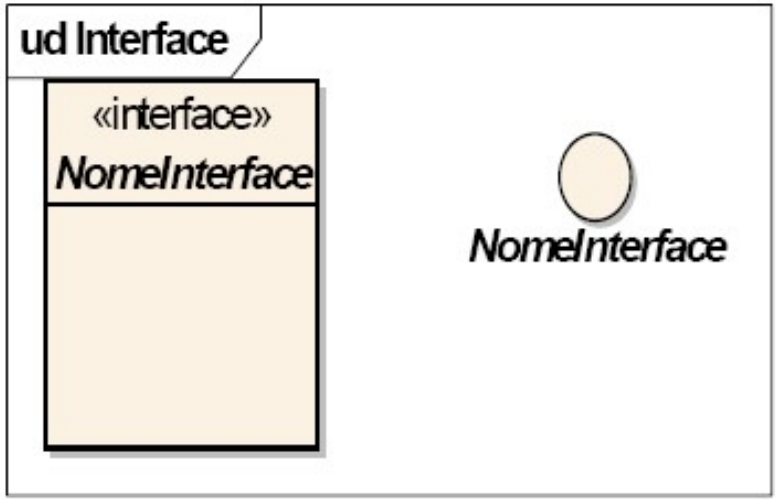
- Exemplo





UML – ITENS ESTRUTURAIS
Interfaces

- Coleção de operações (métodos) que especificam os serviços de uma classe ou componente.
- A interface não contém a implementação das operações, mas apenas descreve quais são estas operações
- Através das interface é possível diminuir o acoplamento entre diferentes partes de um sistema
- Uma interface é realizada por uma ou mais classes

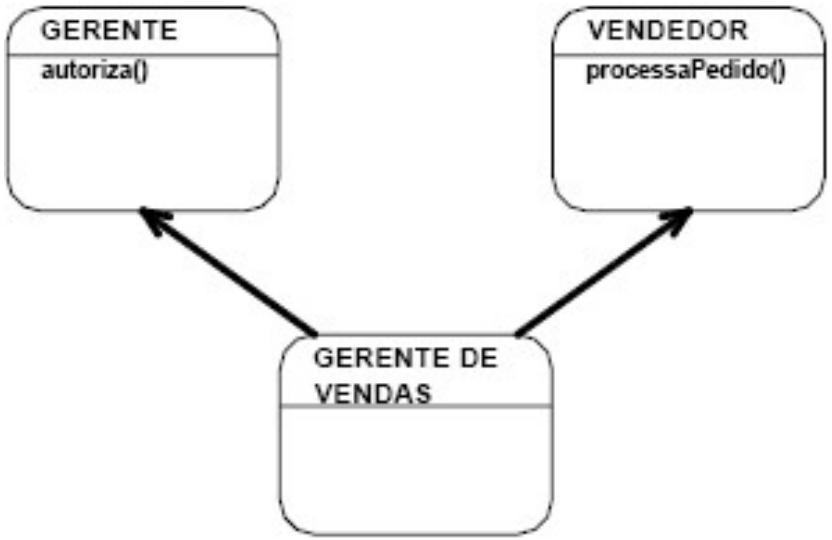


UML – ITENS ESTRUTURAIS
Interfaces

- O conceito de interface é um importante aliado no projeto de software
- Na linguagem Java, este conceito é utilizado para o modelamento da Herança Múltipla

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

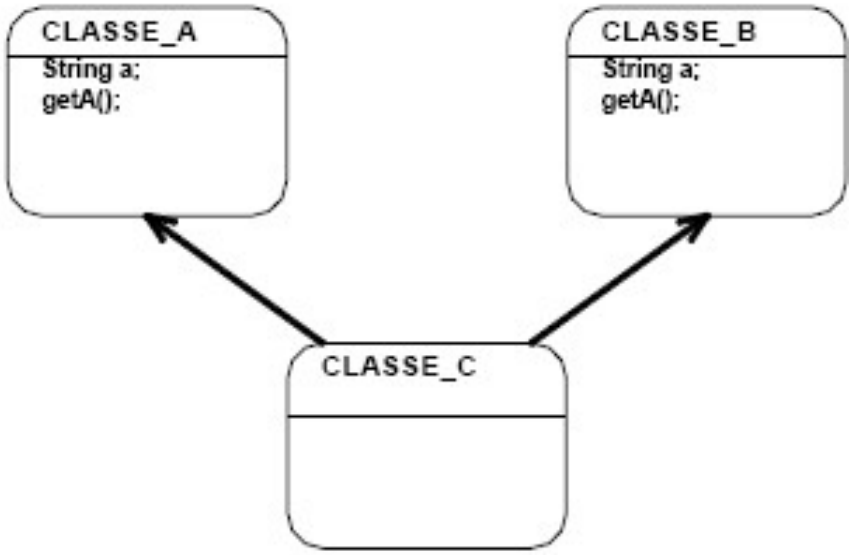
- Existem casos em que uma classe pode herdar o comportamento de mais de uma classe.
- Neste caso temos a herança múltipla, conforme mostrado abaixo



MODELAGEM INCORRETA

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla (java)

- Como implementar a herança múltipla:



- Como implementar a herança múltipla:
- No exemplo acima, qual cópia do atributo **a a classe CLASSE_C** vai herdar? Qual método **getA() vai utilizar?**
- Java resolve este problema utilizando o conceito de “INTERFACES”

Interfaces – Herança Múltipla (java)

- Um método possui duas partes: sua assinatura e sua Implementação
- Java não suporta a herança múltipla explicitamente, mas possui meios para que os efeitos da herança múltipla seja realizada de forma indireta utilizando o conceito de INTERFACES
- Através deste conceito uma classe pode herdar as assinaturas dos métodos, mas não a sua implementação.
- A implementação, deve por sua vez, ser definida na subclasse.
- Desta forma uma interface possui apenas um conjunto de métodos

Classes – Herança Múltipla (Java)

- Através da herança múltipla novos métodos,de diferentes classes, podem ser agregados a uma subclasse
- A herança através de interface não possibilita a reutilização do código, visto que o método herdado deve ser implementado para cada subclasse.
- ATRIBUTOS EM UMA INTERFACE
  - Em uma interface os atributos são implicitamente declarados como static e final.
- MÉTODOS EM UMA INTERFACE
  - Todos os métodos são abstratos, não sendo necessário a palavra “abstract”

Interfaces – Herança Múltipla

- Na interface todos os métodos são abstratos e não possuem implementação apenas sua assinatura.
- A uma classe pode utilizar mais de uma interface em sua definição
- A interface representa um comportamento que a classe que a implementa irá obrigatoriamente possuir.

Interfaces – Herança Múltipla (Java)

- Uma INTERFACE é definida através da palavra “interface” conforme mostrado a seguir:

  [public] interface B extends A

- Neste caso A deve ser outra interface.

  Exemplo – Definição da INTERFACE IGerente

```
public interface IGerente{
    public void demitir(Empregado e);
    public boolean concedeAumento();
    public boolean contratar(Empregado e);
 }
```

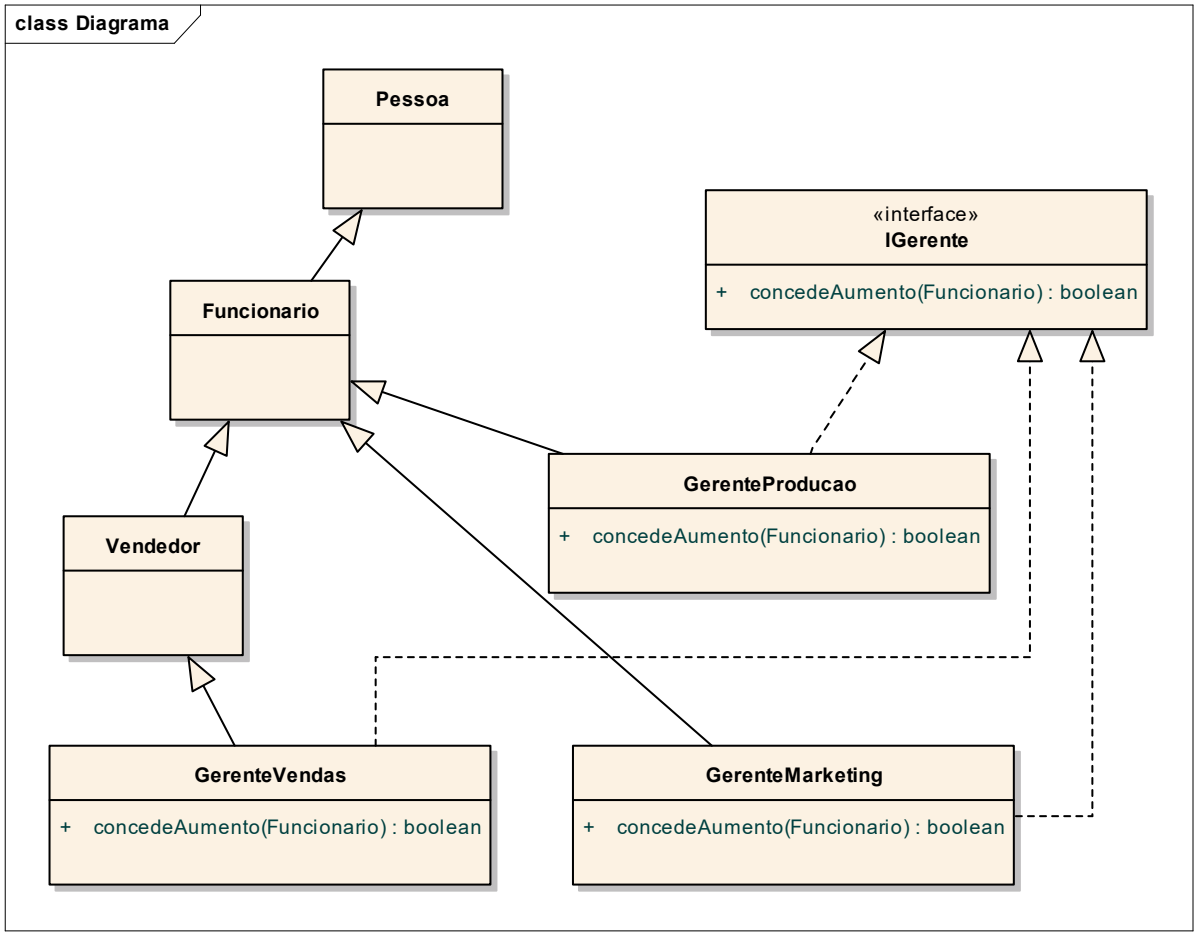
  - O gerente de vendas, bem como qualquer outro tipo de gerente, realiza as operações demitir e contratar, porém cada um a seu modo

- A indicação da herança múltipla é feita da seguinte forma:

  [public] [modTipo] class B [extends A] implements C,[D,E,F..]

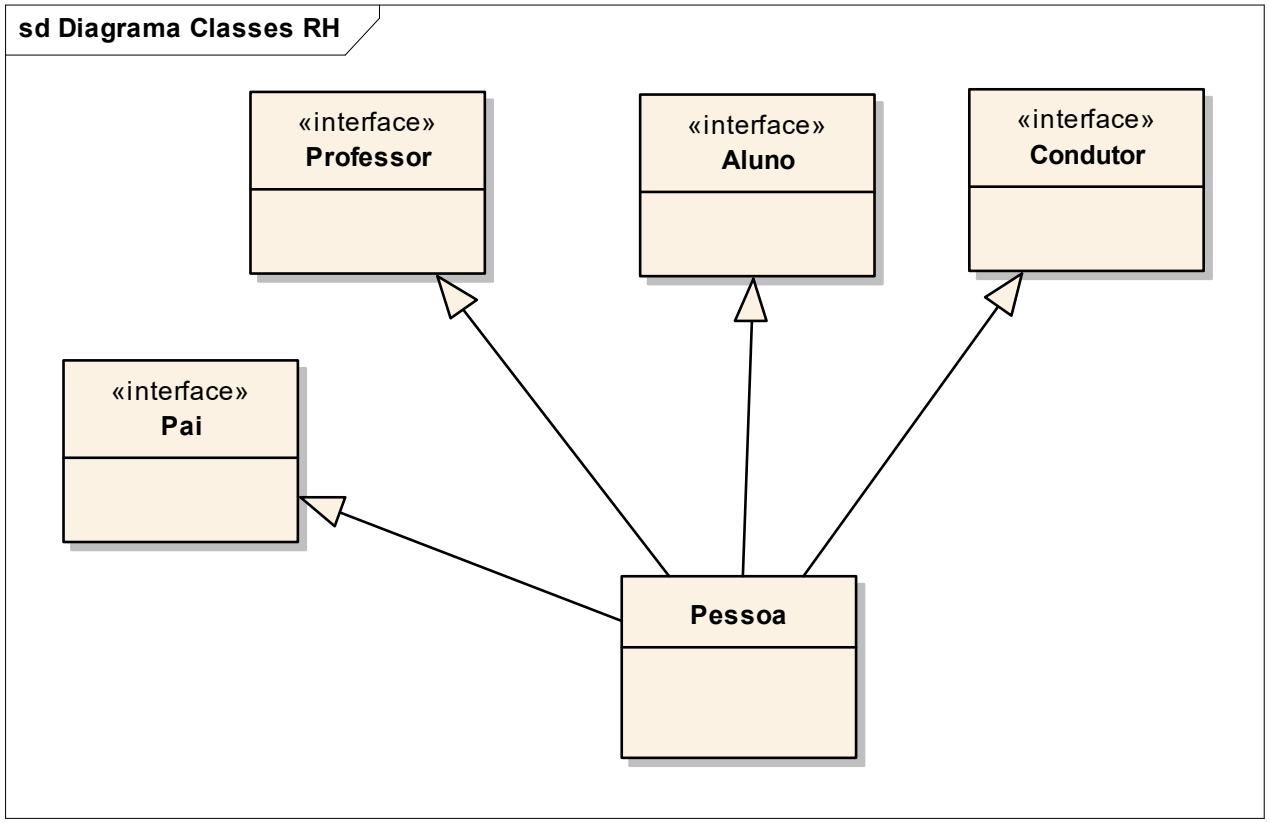
Interfaces – Comportamento Padrão

- Exemplo de Uso de Interface
- A interface garante que um conjunto de classes contenha um comportamento padrão (método), porém com as particularidades necessárias
- Cada tipo de Gerente, pode "concederAumento" porém segundo seus critérios particulares
- A interface garante que todos contenham o método



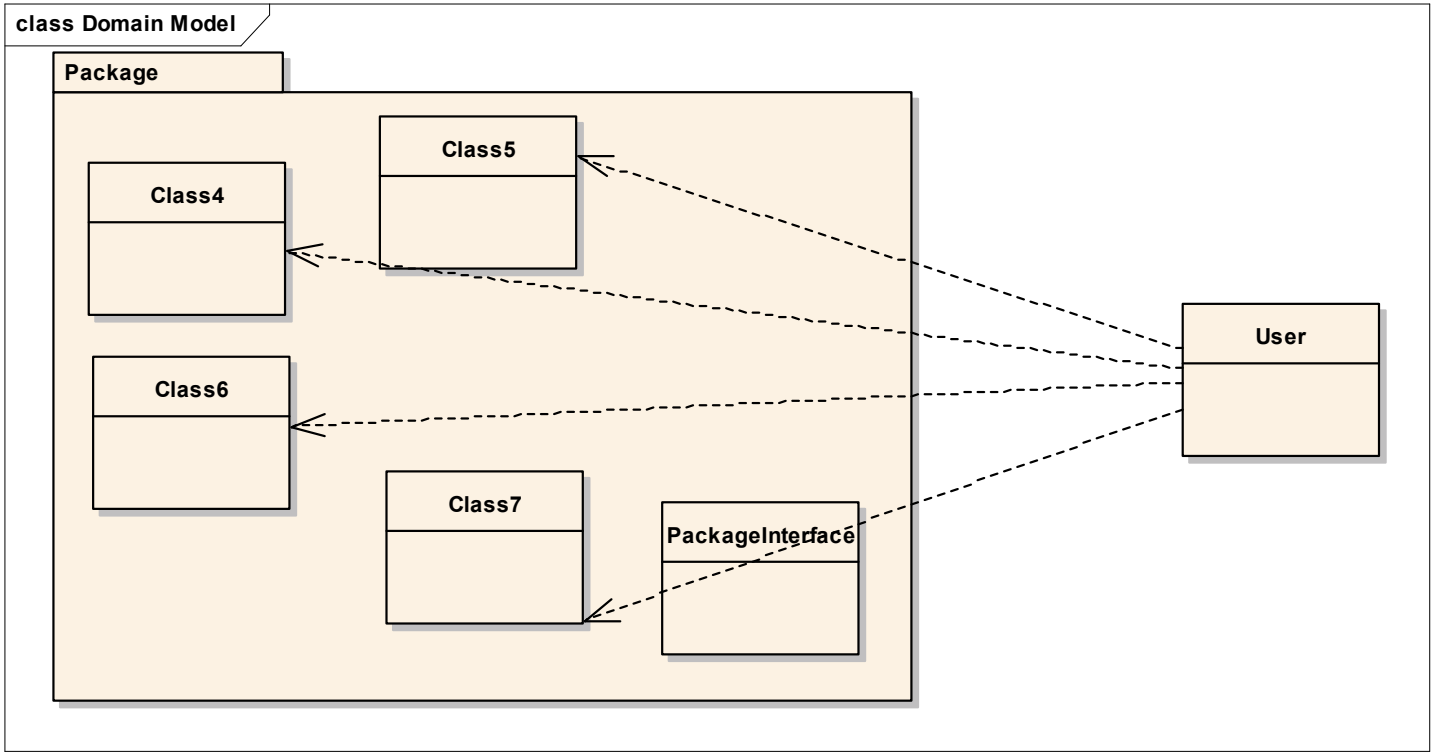
Interfaces – Comportamento Padrão

- Exemplo de Uso de Interface
- A classe Pessoa pode representar diversos papéis. Cada papel associa um conjunto de comportamentos (métodos)
- O que garante a associação é a interface



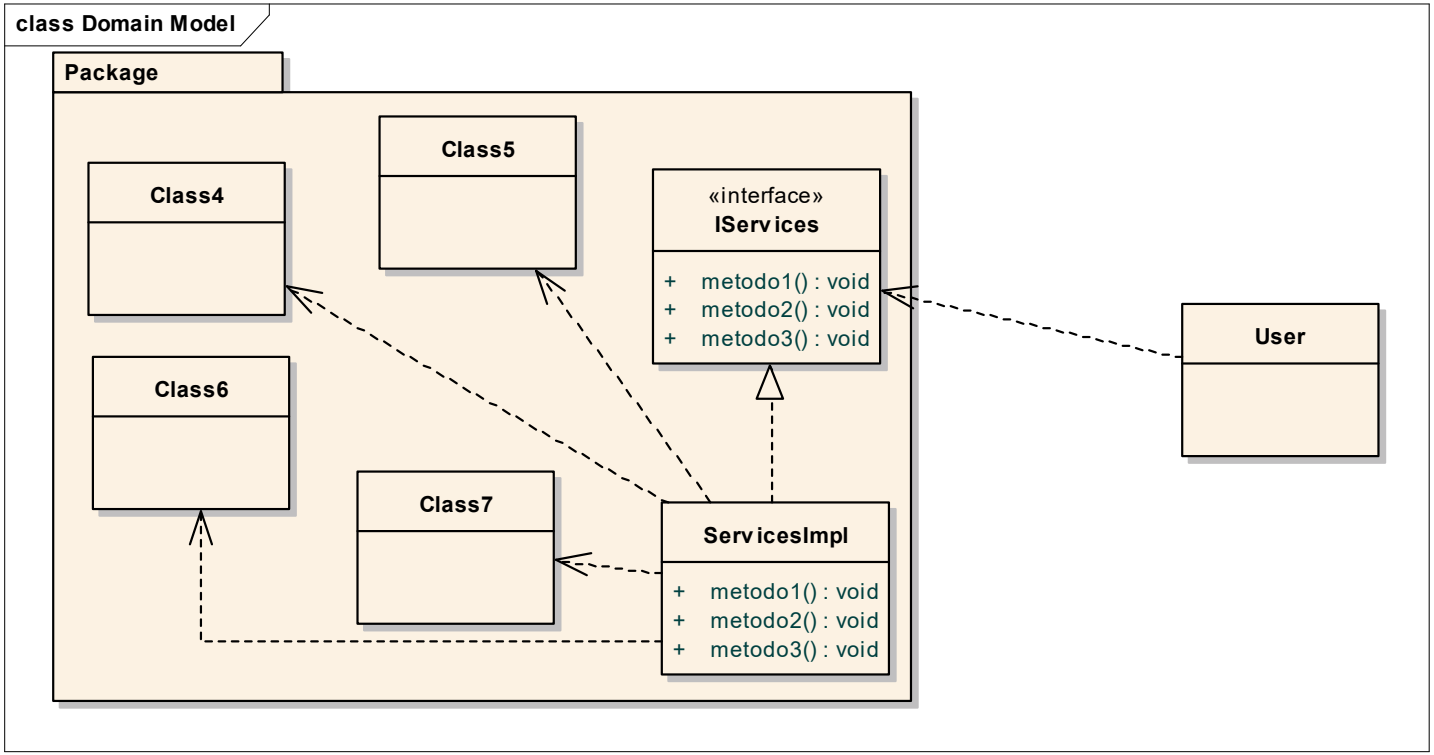
Interfaces – Herança Múltipla

- Redução do Acoplamento
  - O diagrama abaixo mostra uma classe que utiliza outras classes
  - Neste caso existe um alto acoplamento entre as mesmas



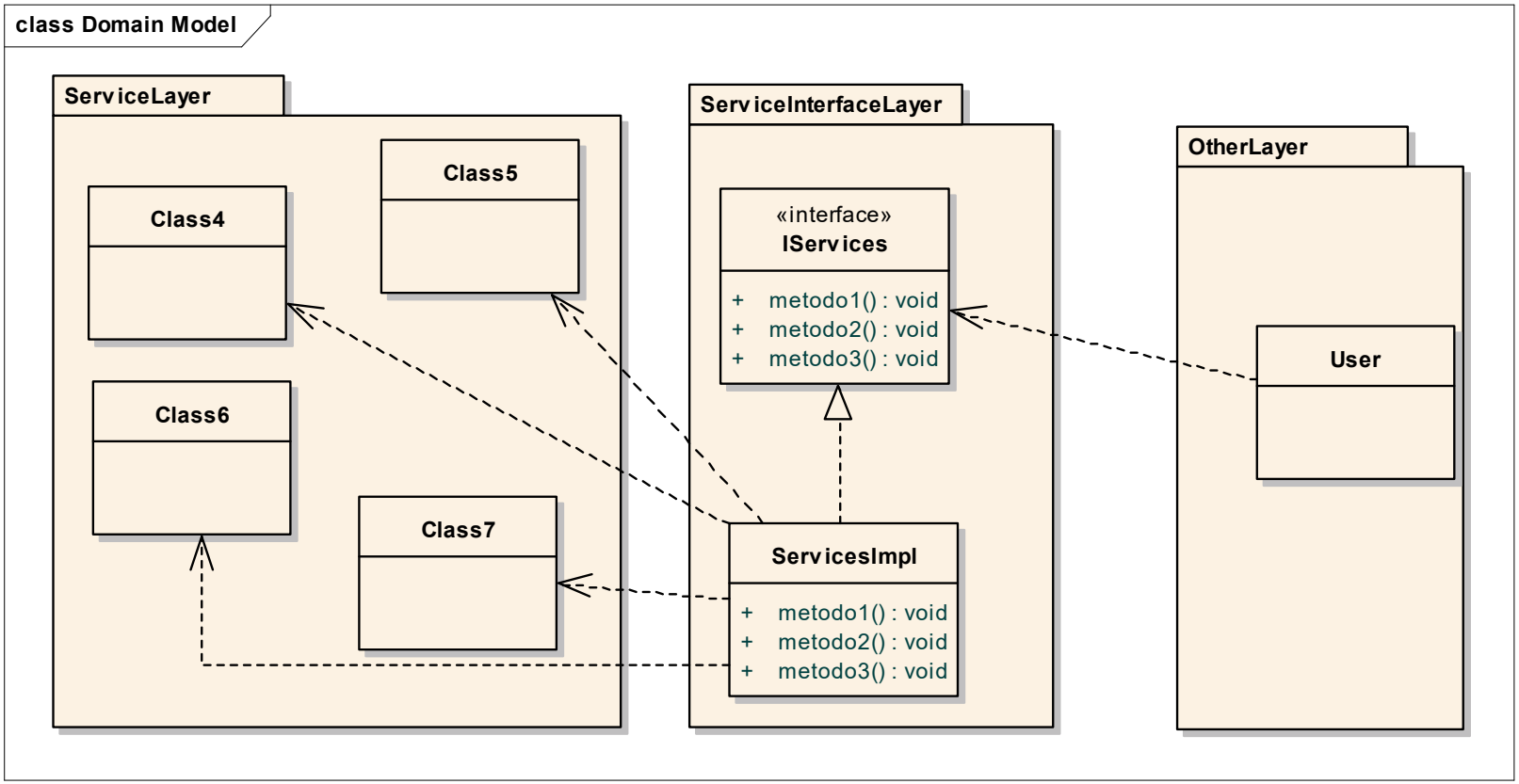
UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

- Redução do Acoplamento
  - A partir do uso da uma Interface é possível reduzir o acoplamento
  - O único ponto de contato entre os módulos neste exemplo é a interface
  - Caso a interface seja mantida (assinatura dos métodos) os módulos são independentes entre si.



Interfaces – Herança Múltipla

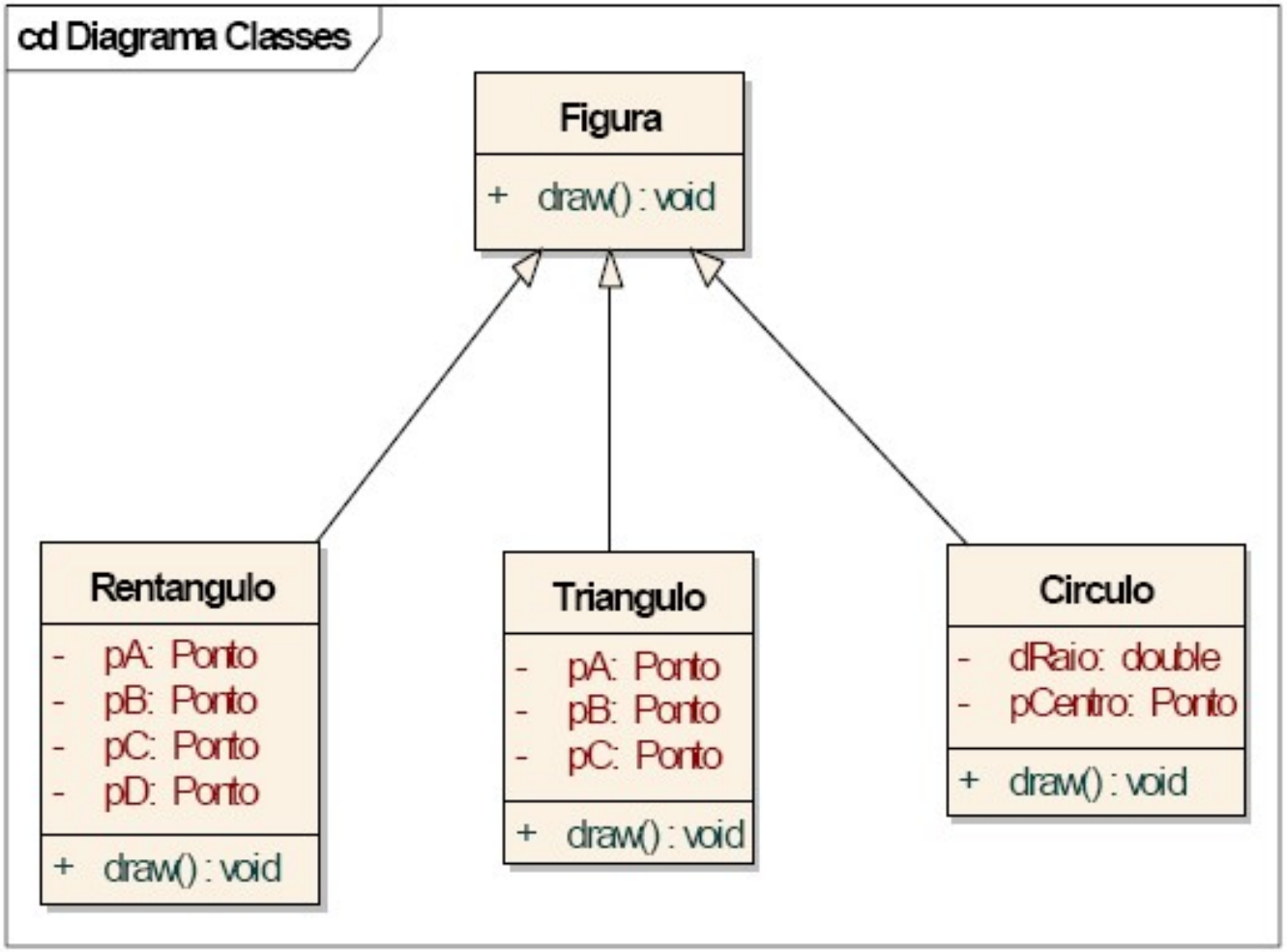
- Redução do Acoplamento
  - A mesma informação, porém organizada de uma diferente forma



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

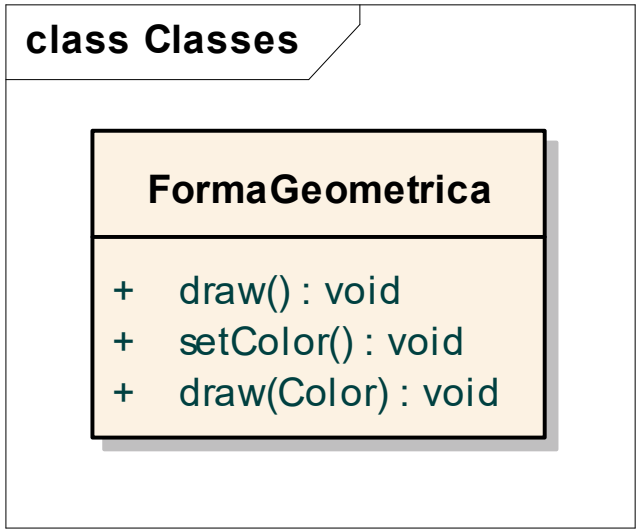
POLIMORFISMO - Exemplo

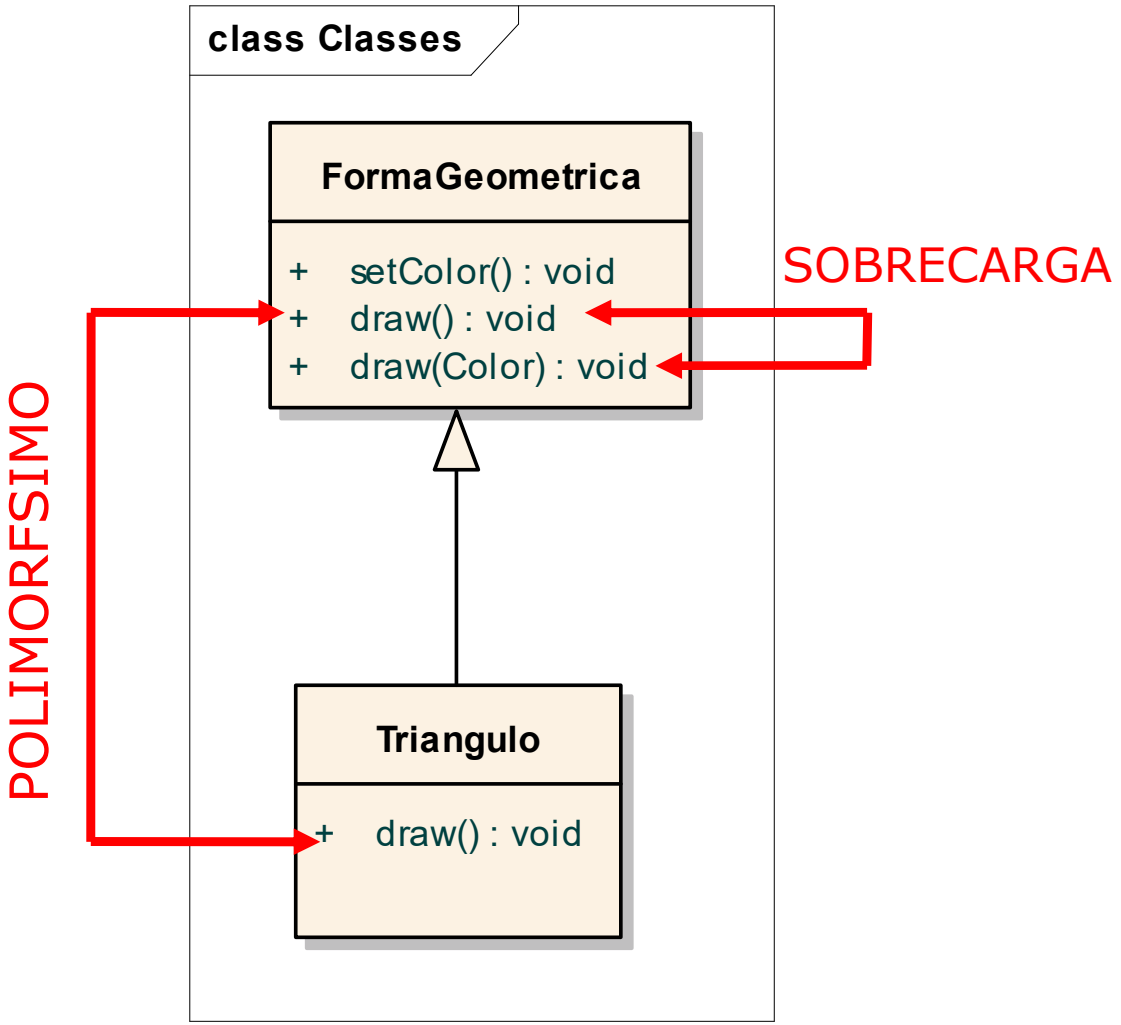


SOBRECARGA(Overload)

- 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 raio 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.

Modularização
Exemplo

- Exemplo de uso de pacotes na organização