

Programação Orientada a Objetos II

- POOII

Revisão 0.0

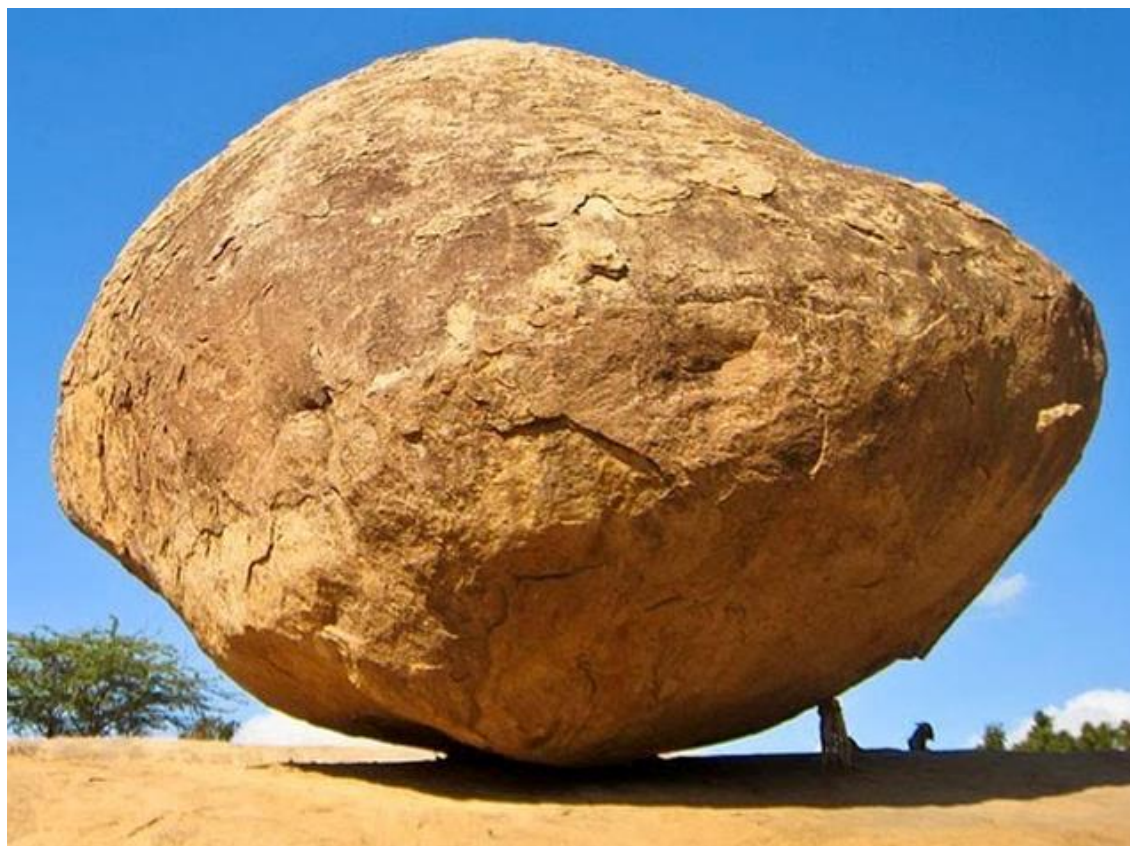
Felipe Frechiani de Oliveira

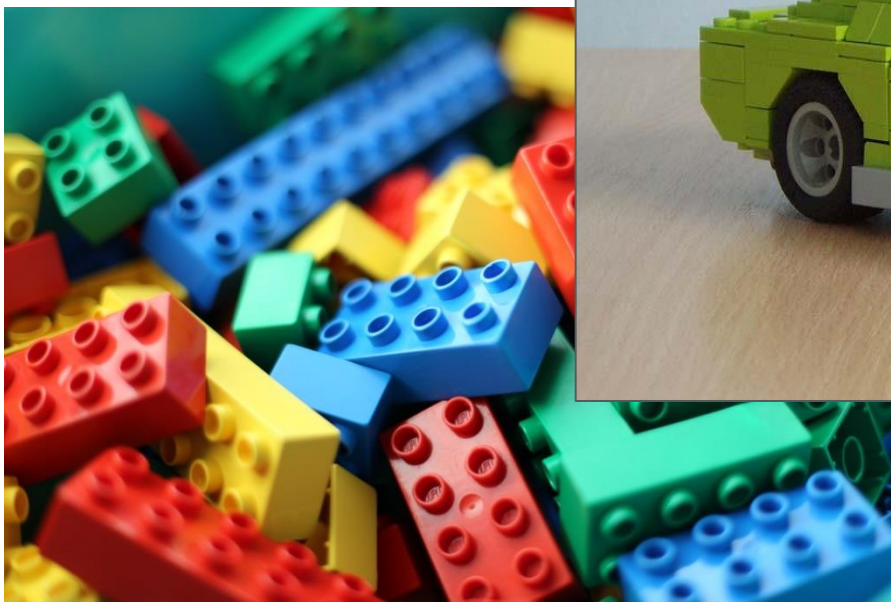
Agenda



- Conceitos de OO
 - JAVA
 - UML
-

Orientação a Objetos





Objetivos da Orientação a Objetos

- Reduzir a complexidade;
 - Aumentar a produtividade de software: reutilização de código e classes;
 - Extensão de biblioteca de classes:
 - Utilização de subclasses (herança);
 - Implementação de novos métodos;,,
 - Redefinição de métodos (polimorfismo)
-

Orientação a Objetos

O mundo real é algo extremamente complexo;

A Orientação a Objetos tenta gerenciar a complexidade dos problemas do mundo real:

- Abstraindo conhecimento relevante;
- Encapsulando o conhecimento dentro de objetos;

Programação Orientada a Objetos

Princípios básicos para gerenciar a complexidade:

Abstração;

Encapsulamento;

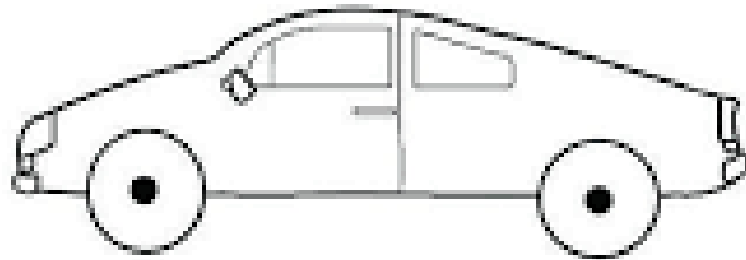
Modularidade;

Hierarquia.

Abstração



Abstração de conceitos relevantes



Abstração

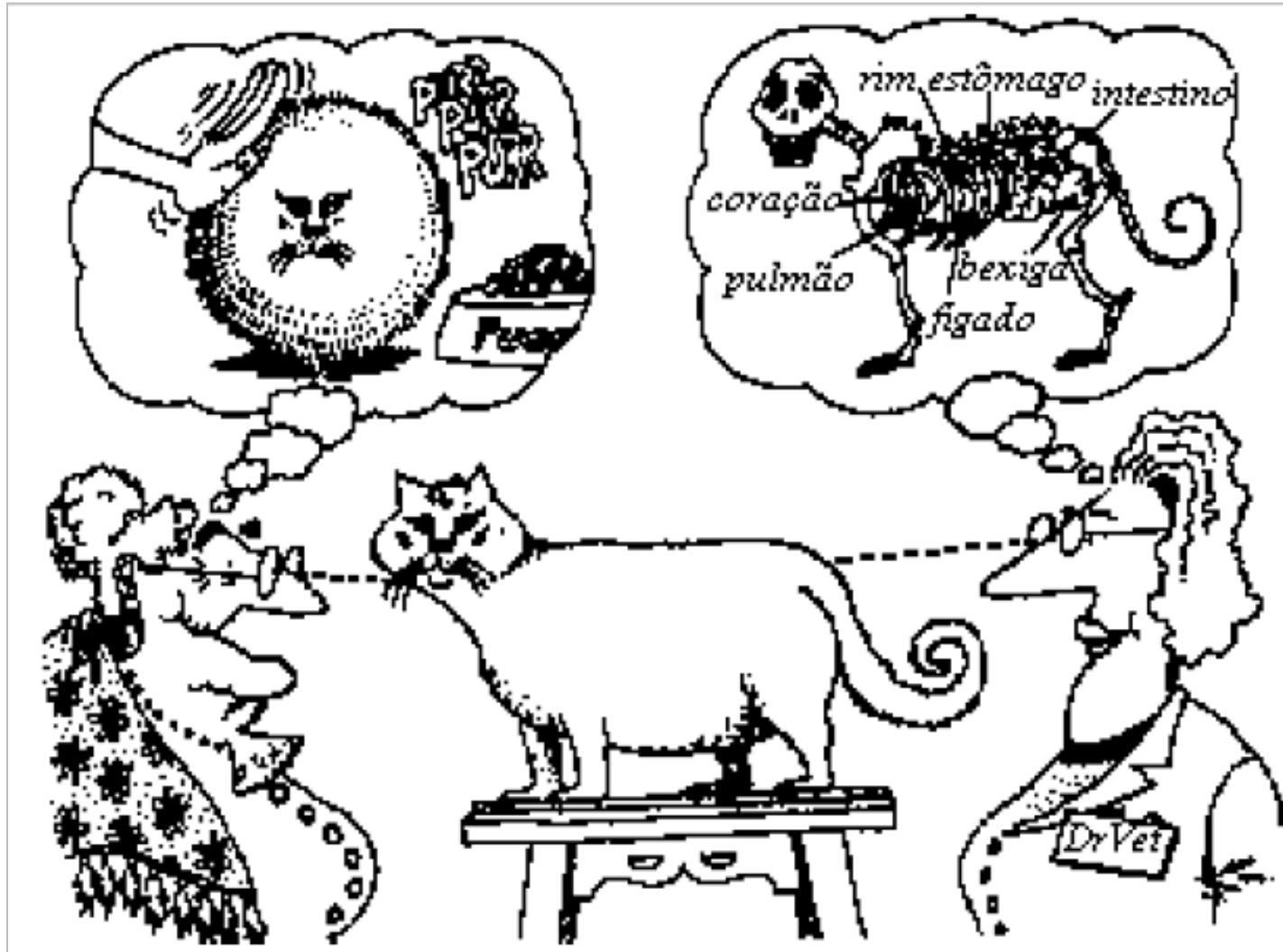
Definição:

“É o princípio de ignorar aspectos não relevantes de um assunto, segundo a perspectiva de um observador, tornando possível uma concentração maior nos aspectos principais do mesmo.”

Características relevantes?



Abstração



Encapsulamento

No mundo real, um objeto pode interagir com outro sem conhecer o funcionamento interno;



Encapsulamento

Definição:

“Consiste na separação dos aspectos externos de um objeto, acessíveis por outros objetos, abstraídos de seus detalhes internos de implementação, que ficam ocultos dos demais objetos”

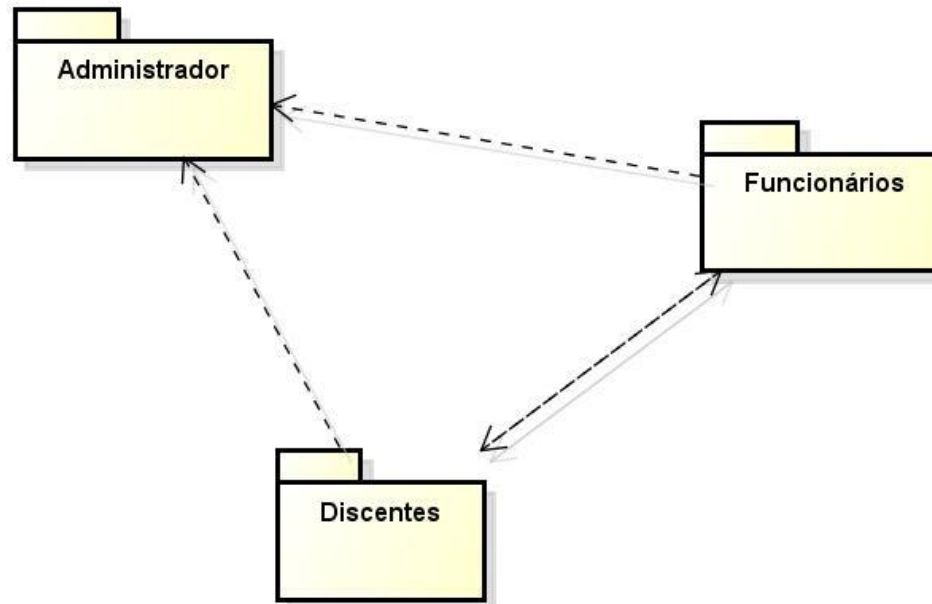
“A interface de comunicação de um objeto deve ser definida de forma a revelar o menos possível sobre o seu funcionamento interno”

Modularidade

Definição:

*“É uma propriedade de sistemas serem decompostos em um conjunto de módulos **coesos** e fracamente **acoplados**”*

- Modularidade é crucial para se obter reusabilidade e extensibilidade;



Coesão e Acoplamento

Em um sistema é desejável uma alta coesão e um baixo acoplamento.

Coesão – cada parte tem somente uma responsabilidade

Baixo acoplamento – baixo grau de dependência

Mecanismo de Estruturação

Utilizado para lidar com a complexidade dos sistemas orientados a objetos;

Tipos de mecanismos de estruturação:

- Ligações e Associações;
 - Composição ou Agregação;
 - Generalização e Especialização.
-

Associações

Ligações e associações são meios de se representar relacionamentos entre objetos e entre classes, respectivamente;

Uma ligação é uma conexão entre objetos

Ex.: “*o empregado João trabalha no departamento RH*”

Existe uma ligação entre os objetos “João” e “Departamento RH”

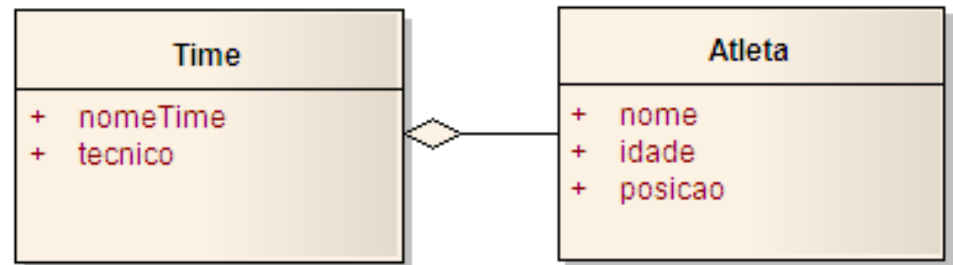
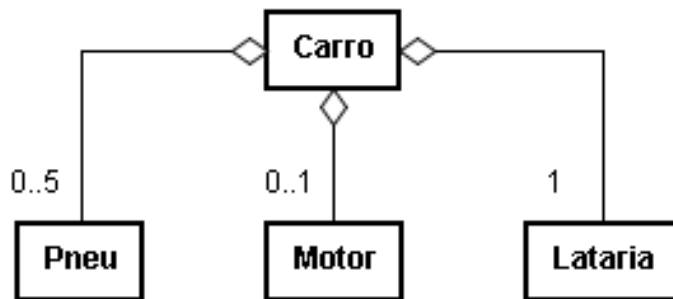
A classe Empregado está associada com a classe Departamento.



Agregação – Relação Todo-parte

- Os objetos contidos podem existir sem serem parte do objeto que os contém.
- Uma das classes é uma parte ou está contida em outra

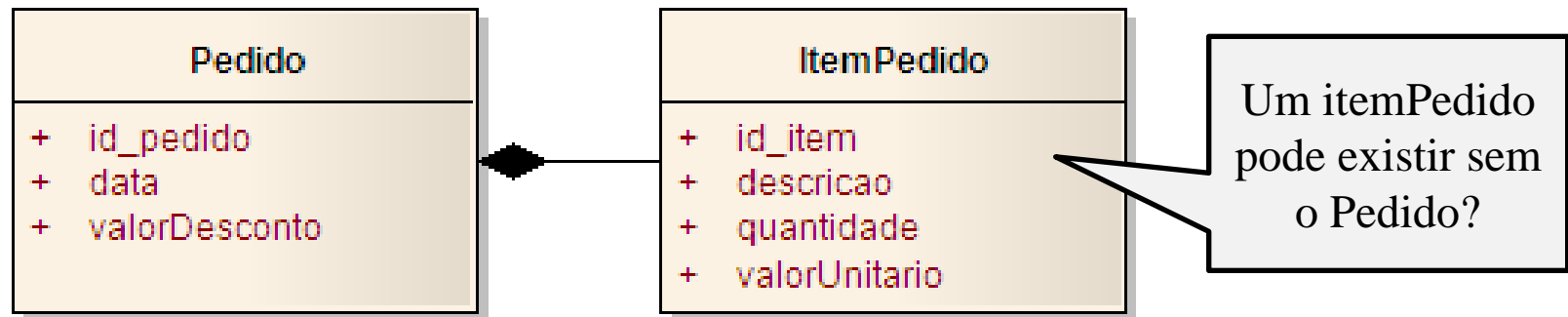
Ex.: O motor, pneus e outras peças dos carro podem existir sem que o carro exista.



Composição

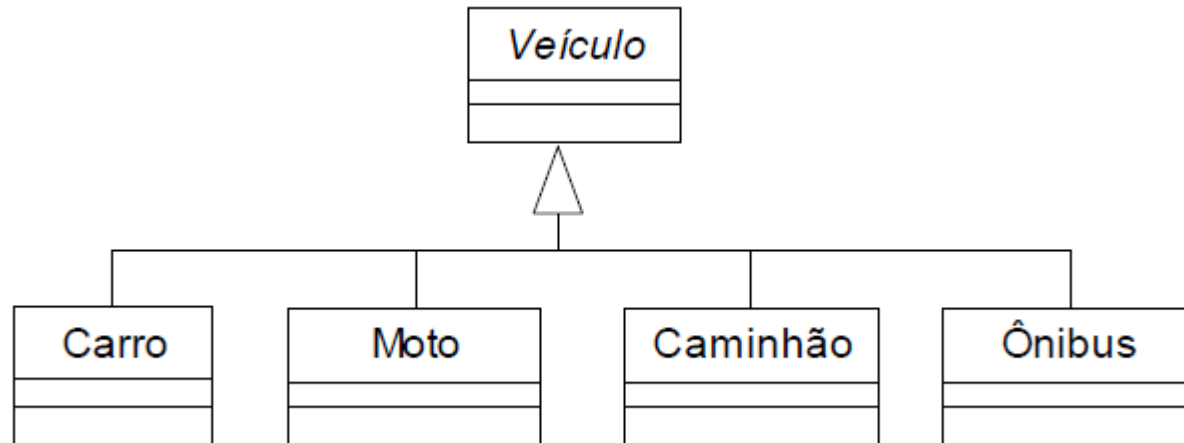
Uma parte constituinte pode pertencer a no máximo uma montagem;

Se o todo deixar de existir a parte também irá deixar de existir



Especialização

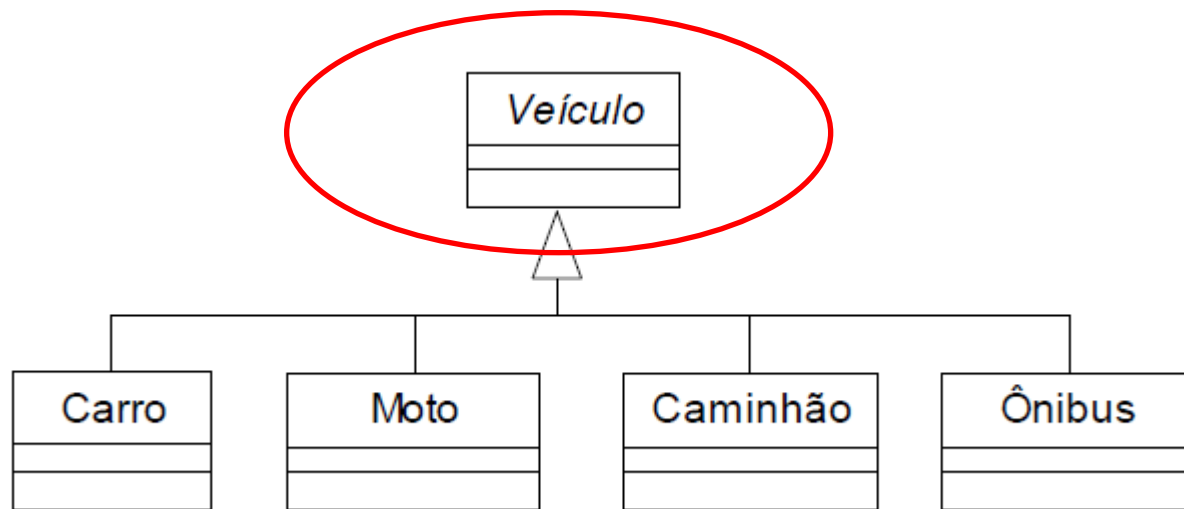
Um conceito geral pode ser especializado, adicionando-se novas características.



Por que especializo?

Generalização

“Operação” inversão da especialização, ou seja, a partir de um conjunto de conceitos, pode-se extrair características comuns, quando generalizadas, formam um conceito geral.



Generalização e Especiaização: Herança

Mecanismo para modelar similaridades entre classes, representando as abstrações de generalização e especialização.

Através da herança, é possível explicitar atributos e serviços comuns em uma hierarquia de classe.

Exercício 1

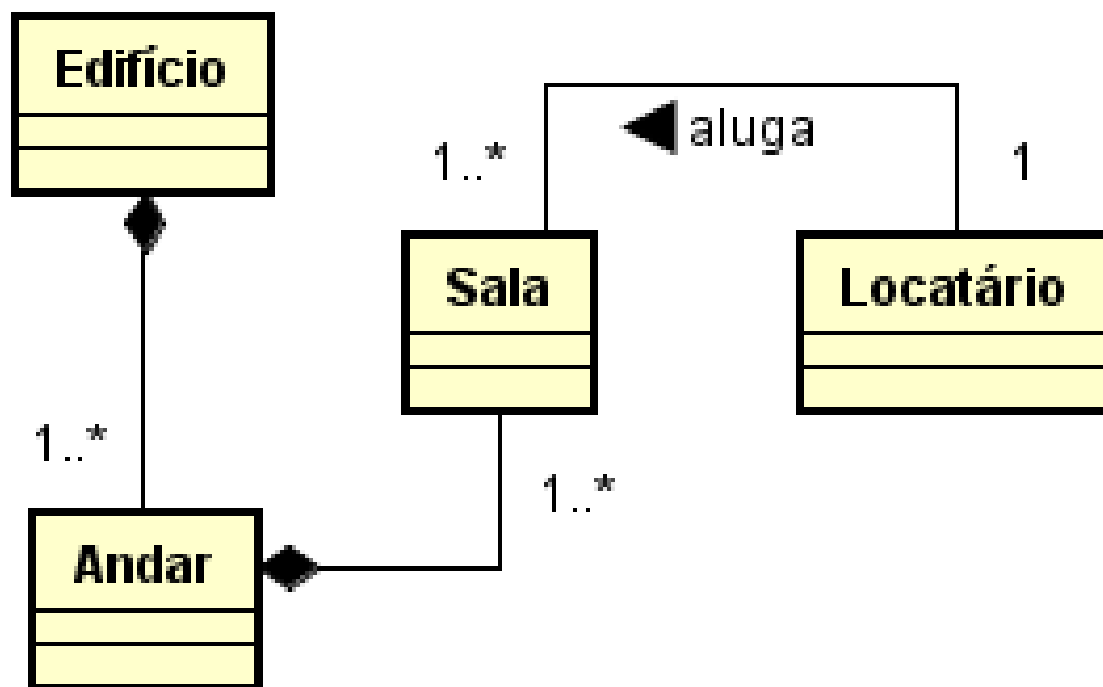
- 1) Faça uma hierarquia para os seguintes conceitos utilizando herança:
 - a) Pessoa, Professor, Aluno, Coordenador e Funcionário;
 - b) Gerente, Líder, Desenvolvedor, Cliente e Pessoa;
-

Exercício 2

Proponha um modelo de classe para um sistema de controle de aluguel de salas de uma imobiliária com os seguintes conceitos: Edifício, Andar, Sala, Locatário.

Exercício 3

Implemente os modelos propostos em java.

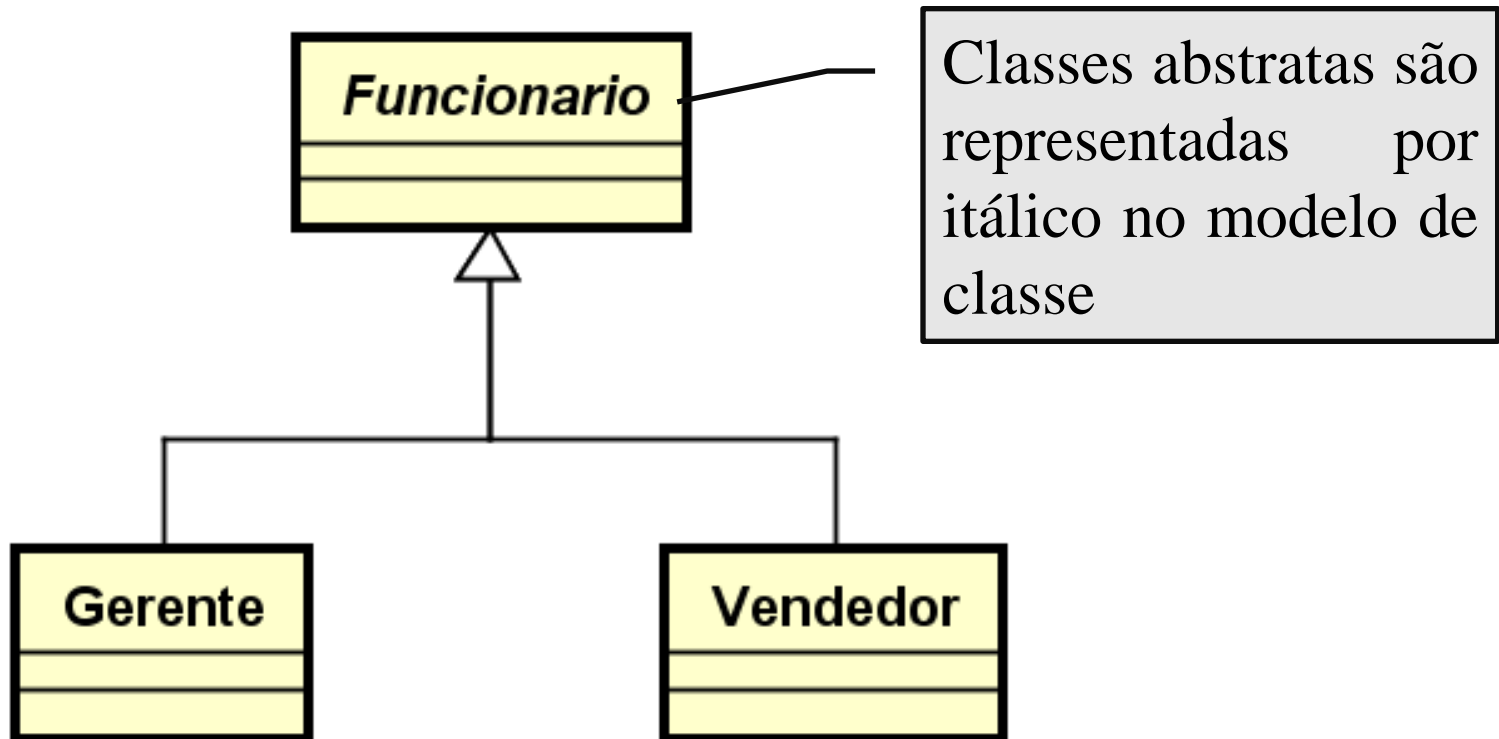


Classe Abstrata

Uma classe abstrata é uma classe que não tem instancias.

Ela serve pra definir um conjunto de métodos e atributos que podem ser herdados e especializados pela classe subclasse.

Classe Abstrata



Classe Abstrata no JAVA


```
abstract class Funcionario {  
  
    protected double salario;  
  
    public double getBonificacao() {  
        return this.salario * 1.2;  
    }  
  
    // outros atributos e métodos comuns a todos Funcionarios  
}
```

```
class Gerente extends Funcionario {  
  
    public double getBonificacao() {  
        return this.salario * 1.4 + 1000;  
    }  
}
```

Interfaces

Uma interface é um contrato que deve ser respeitado por quem a implementa.

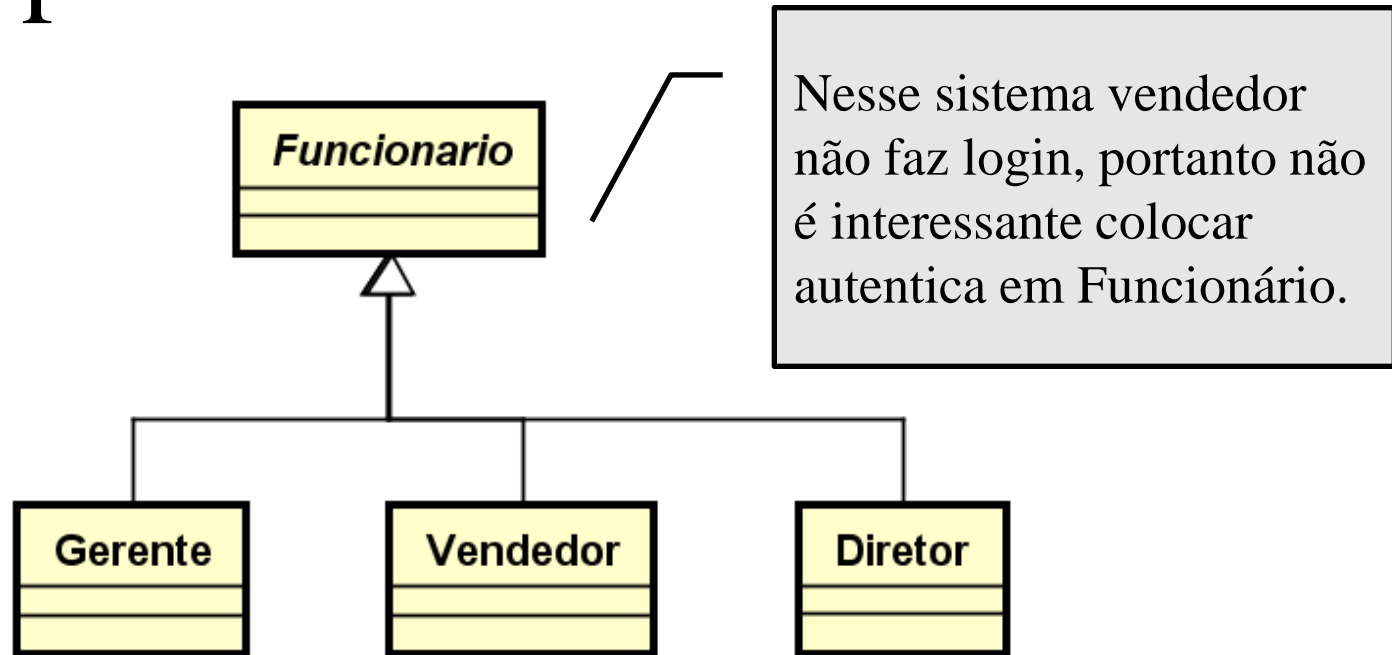
Interface – como representamos



Autenticavel



Exemplo Interface



```
class SistemaInterno {  
  
    void login(Funcionario funcionario) {  
        funcionario.autentica(...); // não compila  
    }  
}
```

Solução 1 – Implementar em ambas as classes

```
class SistemaInterno {  
  
    // design problemático  
    void login(Diretor funcionario) {  
        funcionario.autentica(...);  
    }  
  
    // design problemático  
    void login(Gerente funcionario) {  
        funcionario.autentica(...);  
    }  
  
}
```

Solução 2

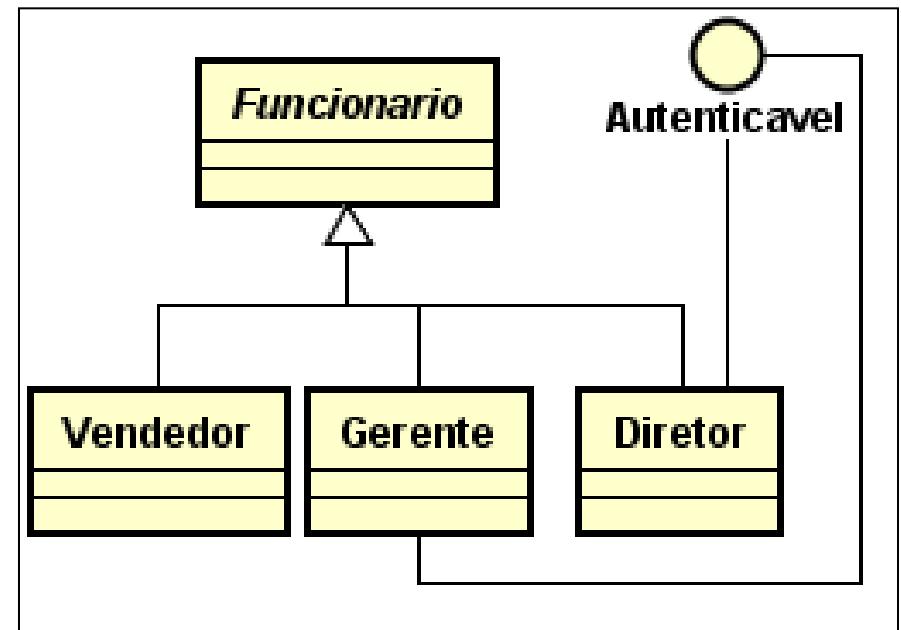
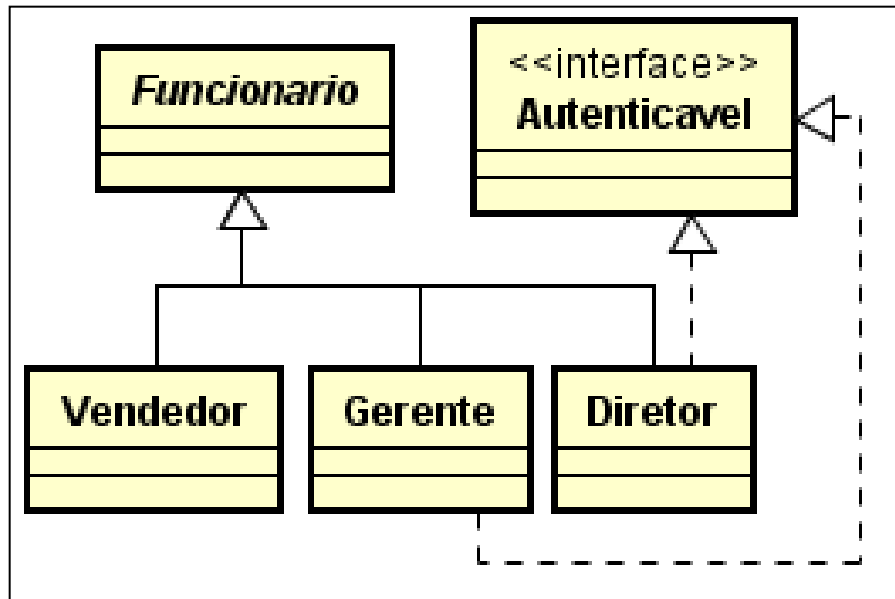
```
interface Autenticavel {  
  
    boolean autentica(int senha);  
  
}
```

```
class Gerente extends Funcionario implements Autenticavel {  
  
    private int senha;  
  
    public boolean autentica(int senha) {  
        if(this.senha != senha) {  
            return false;  
        }  
        return true;  
    }  
  
}
```

Solução 2

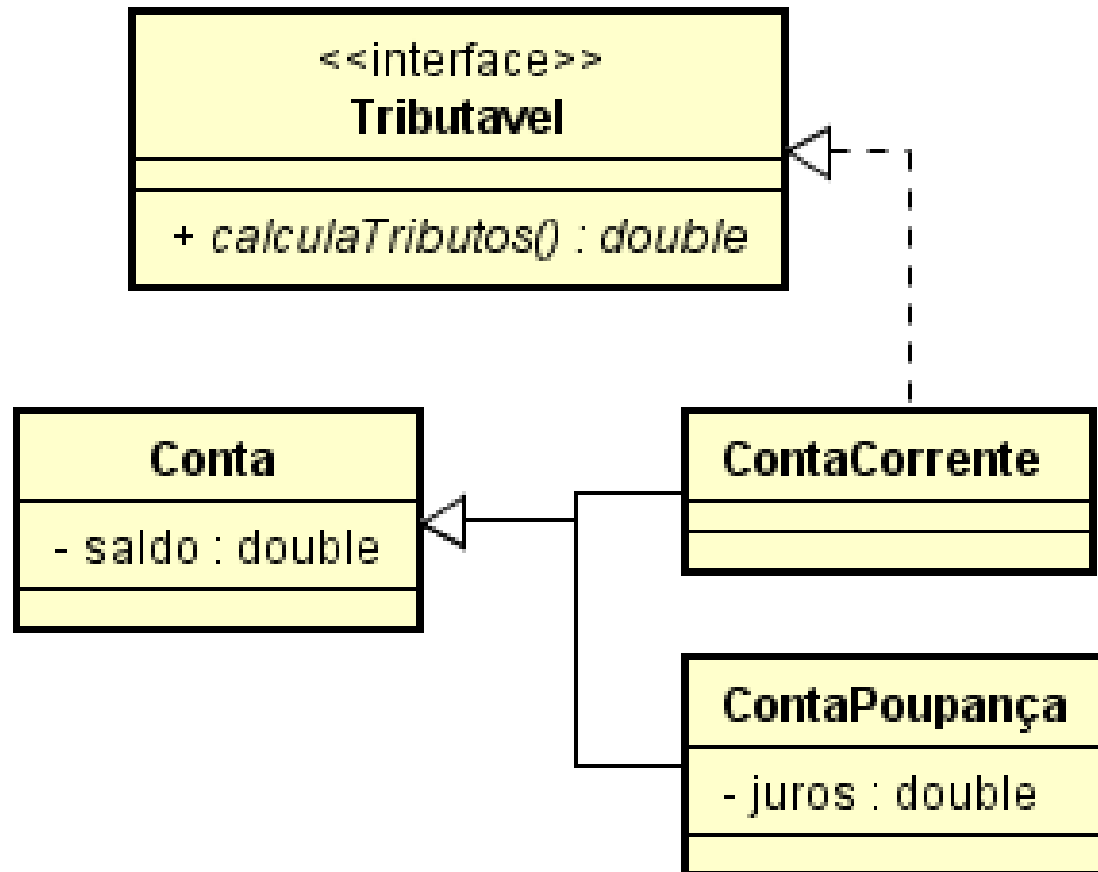
```
class SistemaInterno {  
    void login(Autenticavel a) {  
        boolean ok = a.autentica(senha);  
    }  
}
```

Representação UML do modelo



Exercícios

Implemente o diagrama de classes a seguir em java. Use uma taxa de 10% no calculo do tributo sobre o saldo.



QUESTÃO DISCURSIVA 5

Uma empresa deseja lançar um sistema de comércio eletrônico para vender seus produtos. Essa empresa vende produtos de diversas categorias, como roupas, perfumes e eletrônicos, e aceita diversas formas de pagamento, como cartão de crédito e boleto bancário. No sistema de vendas implementado, cada produto deve ser cadastrado com sua descrição, preço de venda, quantidade em estoque e respectiva categoria. Cada cliente que deseja realizar compras tem de se cadastrar no sistema indicando seu nome, endereço e *e-mail*. Se o cliente for corporativo, deve cadastrar seu CNPJ e, se for individual, seu CPF. O cliente cadastrado pode realizar um pedido de compra dos produtos em estoque na quantidade que desejar. O cliente escolhe uma forma de pagamento disponível e recebe, por *e-mail*, o número do pedido e informações do *status* do pedido. Após a confirmação do pagamento, a loja realiza a entrega dos itens solicitados no endereço do cliente e envia, por *e-mail*, a nota fiscal eletrônica. Tendo em vista que os preços dos produtos podem ser atualizados a qualquer momento, o sistema tem de ser capaz de reemitir uma nota fiscal de um pedido de compra de qualquer produto e respectivo preço na data da compra realizada pelo cliente.

Considerando esse cenário, proponha um Diagrama de Classes, segundo a UML (*Unified Modeling Language*), indicando nome de cada classe, respectivos atributos e relacionamentos entre as classes com as respectivas cardinalidades. Em sua proposta, identifique, pelo menos, um relacionamento de generalização e um relacionamento de composição, não sendo necessário indicar as operações de cada classe. (valor: 10,0 pontos)

- Referências

- <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/classes/classes3.htm>
 - http://download.inep.gov.br/educacao_superior/enade/provas/2014/39_sistemas_informacao.pdf
 - http://download.inep.gov.br/educacao_superior/enade/padrao_resposta/2014/padrao_resposta_sistemas_informacao.pdf
-