

FIA/P GRADUAÇÃO

# DOMAIN DRIVEN DESIGN

Prof. Me. Thiago T. I. Yamamoto

#05 - ENCAPSULAMENTO

# TRAJETÓRIA

---



- ✓ Orientação a Objetos
- ✓ Introdução ao Java
- ✓ IDE e Tipos de Dados
- ✓ Classes, atributos e métodos
- ✓ Encapsulamento

# #05 - AGENDA

---

- Pacotes
- Modificadores de Acessos
- Encapsulamento
- Java Beans

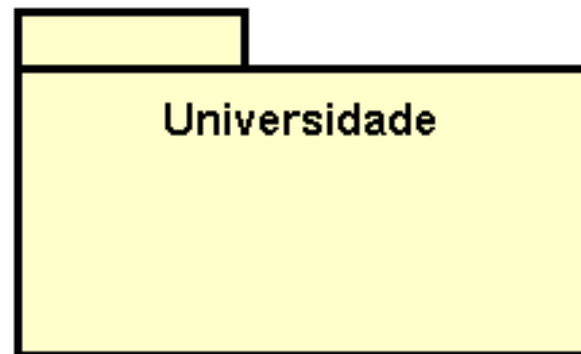




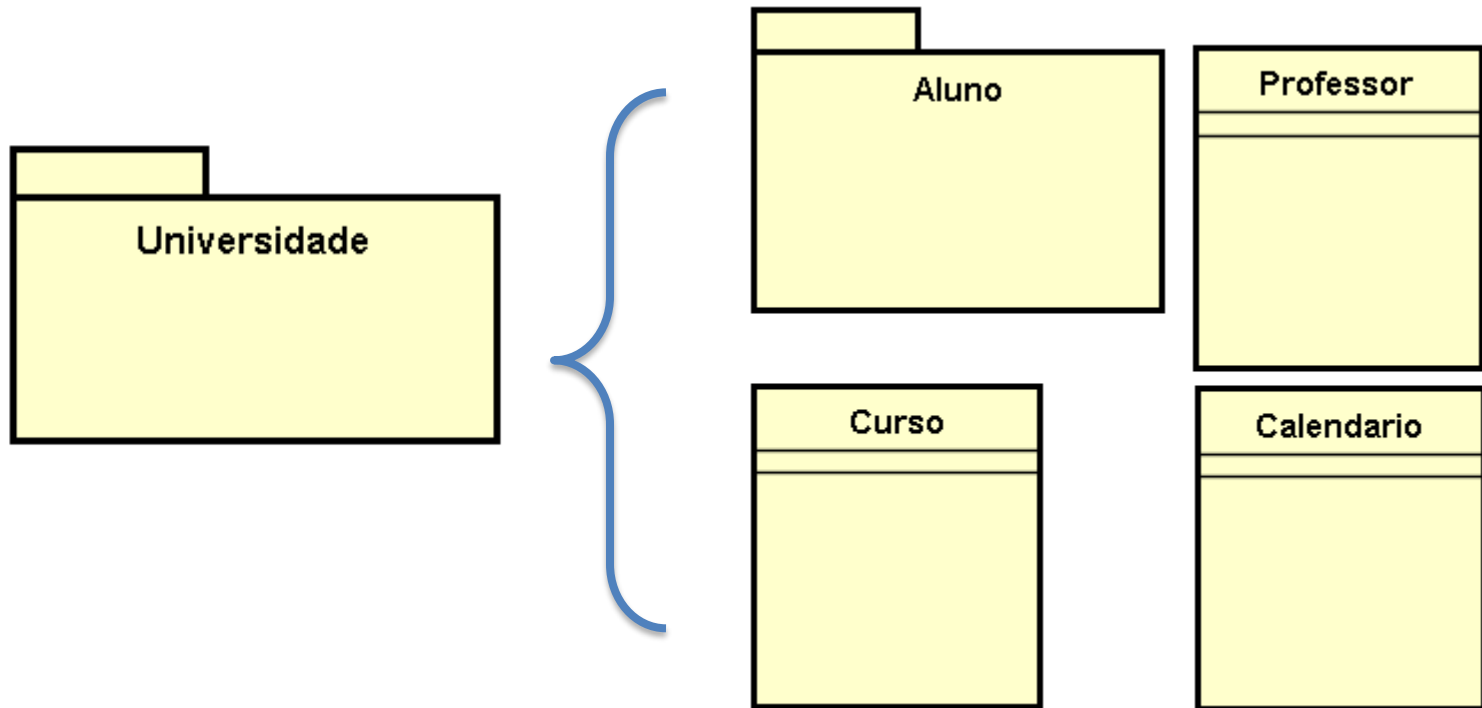
# PACOTES

- Um mecanismo de propósito geral para **organizar elementos em grupos**;
- Um **elemento do modelo** que pode conter outros elementos do modelo;
- O **pacote** pode ser usado:
  - Para **organizar** o modelo em desenvolvimento;
  - Como uma **unidade de gerenciamento de configuração**;

Representação Gráfica:



- Um **pacote** pode conter **classes** e **outros pacotes**:
  - O pacote *Universidade* contém um pacote e três classes:

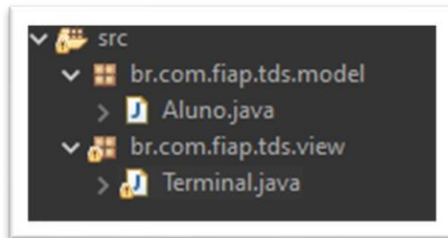
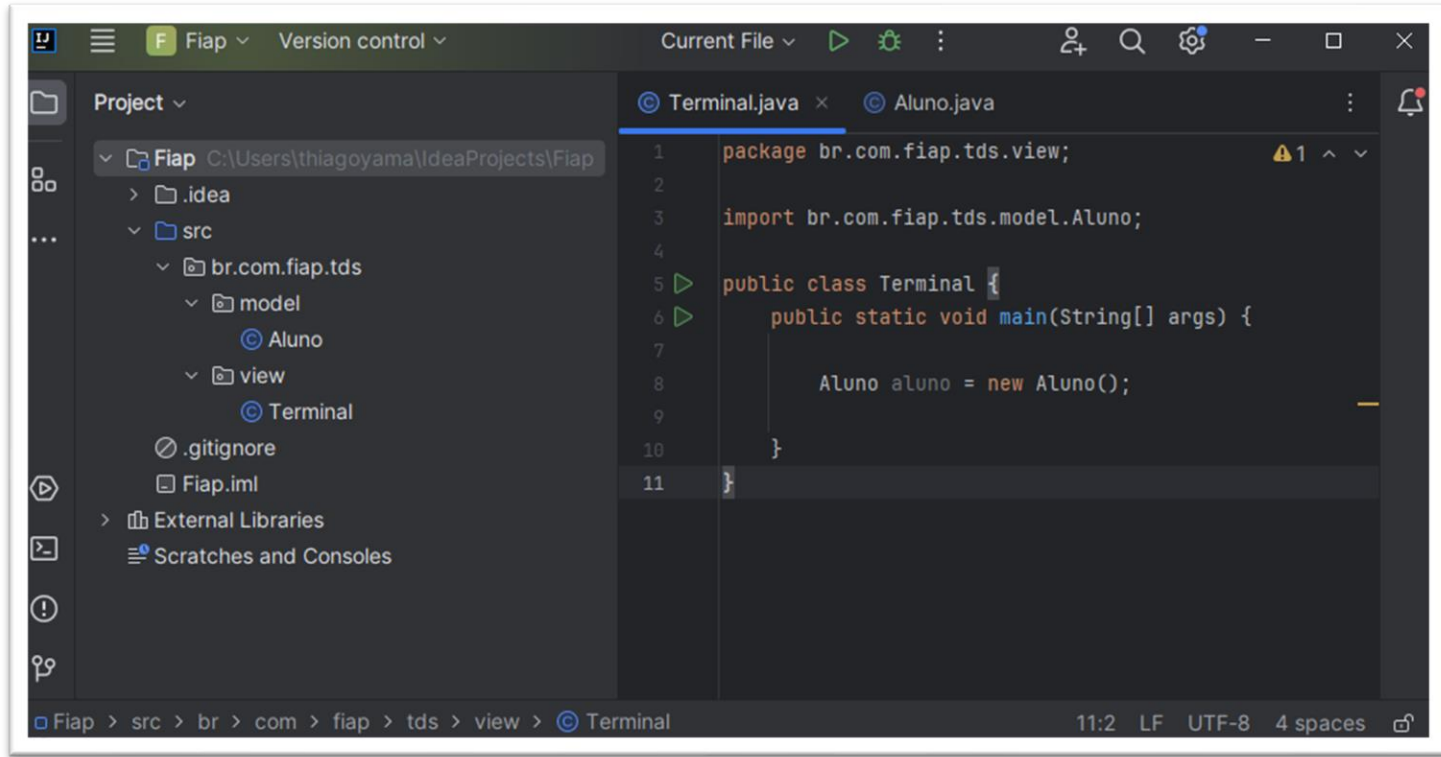


- Além das **classes**, o **Java** provê um recurso adicional que ajuda a modularidade: o **uso de pacotes**;
- **Pacotes** permitem a criação de **espaços de nomes**, além de mecanismos de controle de acesso;
- **Pacotes** são tipicamente implementados como **diretórios**;
- Os **arquivos das classes** pertencentes ao **pacote** devem ficar em seu **diretório**;
- **Hierarquias de pacotes** são construídas através de **hierarquias de diretórios**;

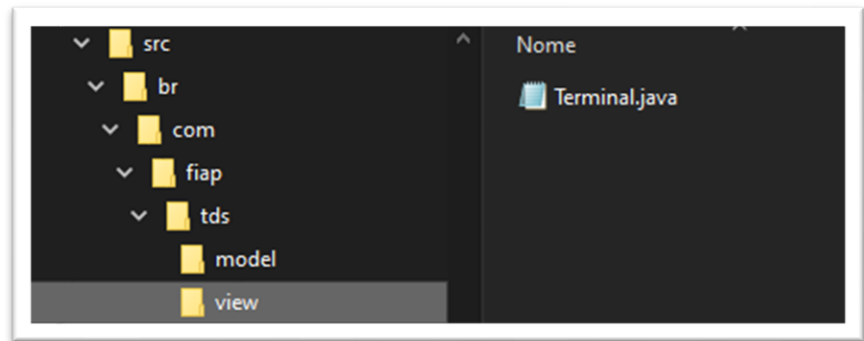


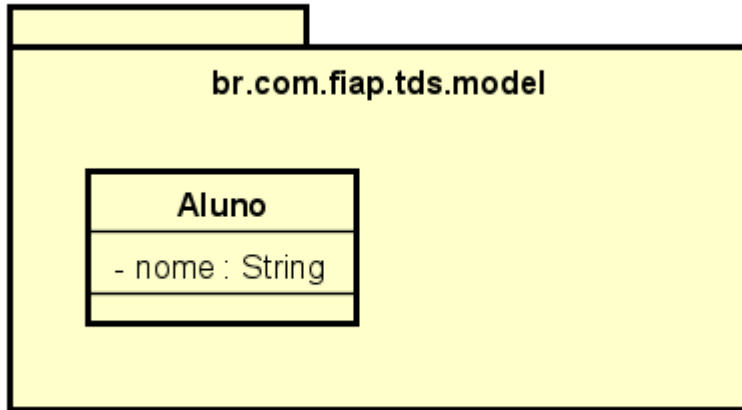


- “Empacotando” uma **Classe**:
  - Para declararmos uma **classe** como pertencente a um **pacote**, devemos:
    - declará-la em um **arquivo dentro do diretório** que representa o pacote;
    - declarar, na **primeira linha do arquivo**, que a **classe** pertence ao **pacote**;
- **Importação de Pacotes**:
  - Podemos usar o **nome simples** (não qualificado) de uma classe que pertença a um pacote se **importarmos a classe**;
  - A importação de uma classe (ou classes de um pacote) pode ser feita no **início do arquivo, após a declaração do pacote** (se houver);
  - As classes do pacote padrão **java.lang** não precisam ser importadas (Ex.: **String**)

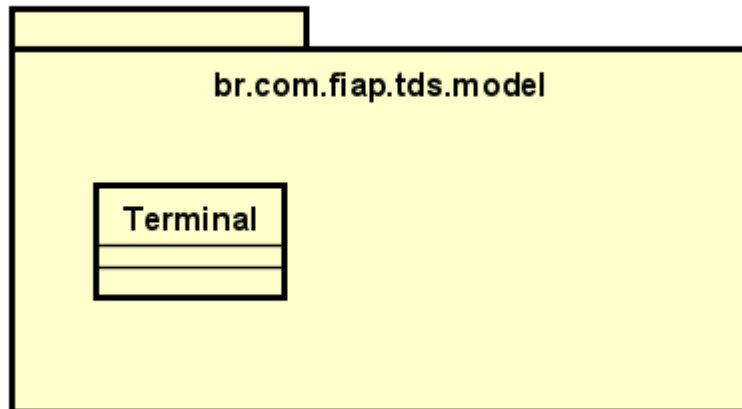


=





```
package br.com.fiap.tds.model;
```



```
package br.com.fiap.tds.view;
```



# MODIFICADORES DE ACESSO

- As linguagens OO disponibilizam formas de controlar o acesso aos membros (atributos e métodos) de uma classe. No mínimo, podemos controlar o que é **público** ou **privado**;
- O **Java** disponibiliza três modificadores de acesso:
  - **private**
  - **protected**
  - **public**
- Quando nenhum modificador é utilizado, dizemos que o membro está com o **nível de acesso default**, também conhecido como **package**;
- Membros **públicos** podem ser acessados por todos, enquanto os **privados** só podem ser acessados pela própria classe;



Símbolo	Palavra-chave	Descrição
-	private	Atributos e métodos são acessíveis somente nos métodos da própria classe. Este é o nível <u>mais rígido</u> de encapsulamento.
~		Atributos e métodos são acessíveis somente nos métodos das classes que pertencem ao pacote em que foram criados.
#	protected	Atributos e métodos são acessíveis nos métodos da própria classe e suas subclasses.
+	public	Atributos e métodos são acessíveis em todos os métodos de todas as classes. Este é o nível <u>menos rígido</u> de encapsulamento.



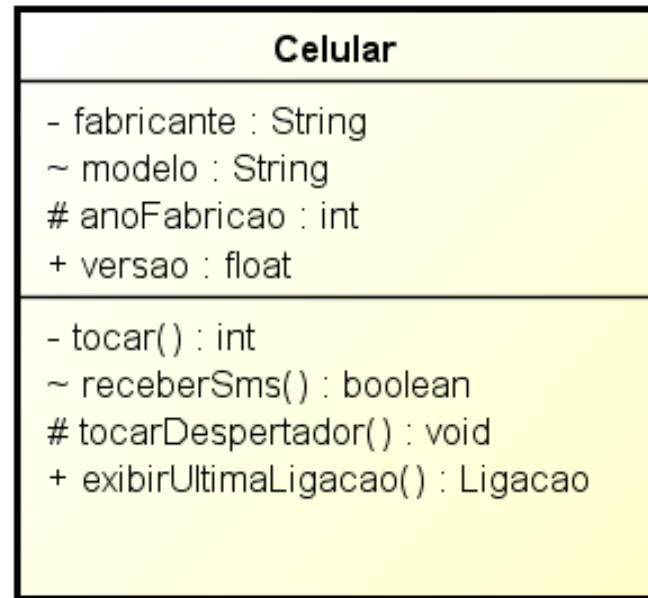
- Os modificadores de acesso podem ser representados no diagrama de classes através dos símbolos:

- (private)

~ (*default*)

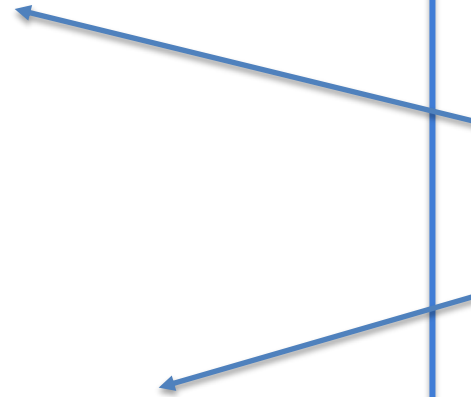
# (protected)

+ (public)



```
private String fabricante;  
String modelo;  
protected int anoFabricacao;  
public float versao;  
  
private int tocar(){ ... }  
boolean receberSms(){ ... }  
protected void tocarDespertador(){ ... }  
public Ligacao exibirUltimaLigacao(){ ... }
```

Celular
- fabricante : String ~ modelo : String # anoFabricacao : int + versao : float
- tocar() : int ~ receberSms() : boolean # tocarDespertador() : void + exibirUltimaLigacao() : Ligacao







# ENCAPSULAMENTO

- É aplicado aos **atributos** e **métodos** de uma **classe**;
- Consiste em **proteger os dados** ou até mesmo escondê-los;
- Para **limitar** ou **controlar** o conteúdo de um atributo, **métodos devem ser utilizados para colocar ou alterar valores** dos atributos de um objeto;
- Para **limitar o acesso a um método**, métodos devem ser utilizados para acessar o método com **visibilidade restrita**;
- O uso de **atributos diretamente** pelos clientes de uma classe é **desencorajado**:
  - Quaisquer **mudanças** na estrutura interna da classe acarretariam em **mudanças nos clientes**;
- Dependendo da **visibilidade**, o acesso aos atributos **não podem ser feito diretamente**;

- **Benefícios:**
  - **Esconde os detalhes** da implementação de uma classe;
  - Força o usuário a **usar um método para acesso aos dados**;
  - Permite definir o modo de **acesso aos dados**:
    - **Leitura**;
    - **Escrita**;
    - **Leitura/Escrita**;
  - **Proteger os dados** que estão dentro dos objetos, evitando assim que os mesmos **sejam alterados de forma errada**;



- O uso de métodos de leitura (get) e escrita (set) visam desacoplar os atributos de uma classe dos clientes que a utilizam;
- No exemplo a seguir, o atributo **idade** está encapsulado:

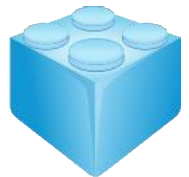
```
public class Pessoa {  
    private int idade;  
  
    public int getIdade(){  
        return idade;  
    }  
  
    public void setIdade(int idade){  
        this.idade = idade;  
    }  
}
```

Pessoa
- idade : int
+ setIdade(idade : int) : void + getIdade() : int

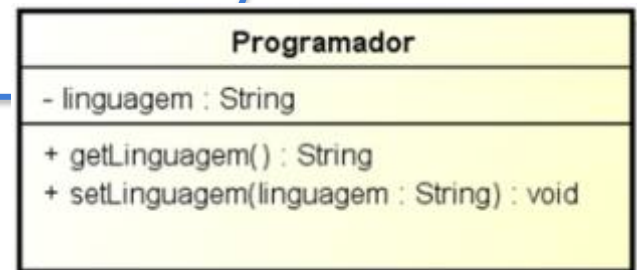
- No exemplo abaixo, o método **formatarTel** está encapsulado:

```
public class Telefone{  
  
    private String ddd, numero;  
  
    public String getTelefoneFormatado(){  
        return formatarTel(ddd,numero);  
    }  
  
    private String formatarTel(String ddd, String numero){  
        return "(" + ddd + ") " + numero;  
    }  
  
}
```

- Os **JavaBeans** são componentes de software projetados para serem reutilizáveis, usados de uma maneira que **permita isolar e encapsular** um conjunto de funcionalidades;
- Podemos definir um **Bean** como uma **classe Java** que segue um **conjunto de convenções de design e nomeação** definidos pela especificação de **JavaBeans** do JSE ;
- Um bean tem como premissa a ideia de **encapsulamento**. Assim sendo suas variáveis devem **obrigatoriamente** ser acessadas através de métodos;
- Outra importante regra refere-se ao **construtor**. Ele deve ser **sem parâmetros**;

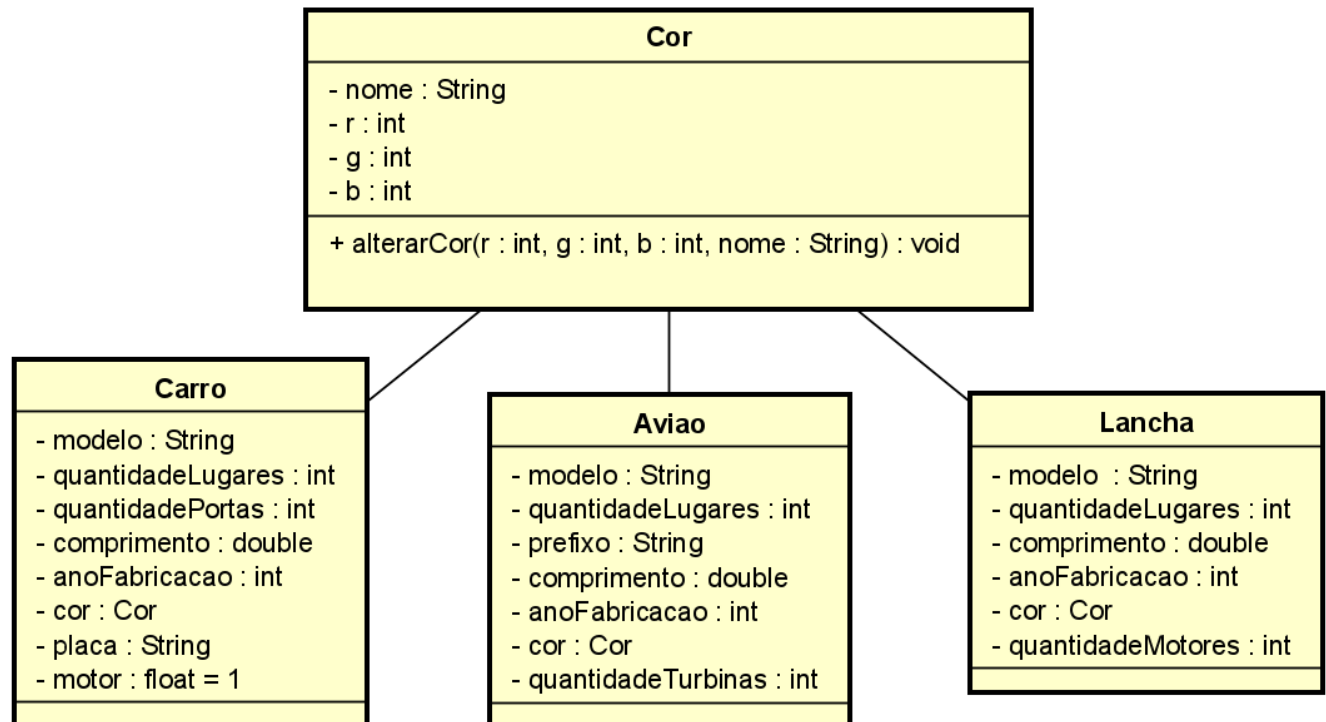


```
public class Programador {  
    private String linguagem;  
  
    public String getLinguagem(){  
        return linguagem;  
    }  
  
    public void setLinguagem(String linguagem){  
        this.linguagem = linguagem;  
    }  
}
```



# PRÁTICA 1

- Crie as **classes** abaixo com os métodos **getters** e **setters**;
- **Instancie** as **classes** em uma classe de teste;





# PRÁTICA 2

---



- Escreva uma classe chamada **Data** que contenha três atributos do tipo `int` chamados dia, mes e ano.
  - Crie um método para retornar a data como string no formato: `dd/mm/aaaa`;
  - Crie um método para validar o dia. Ele deve estar entre 1 e 31. Caso contrário inicializar o atributo com o valor 1;
  - Crie um método para validar o mês. Ele deve estar entre 1 e 12. Caso contrário inicializar o atributo com 1;
  - Crie um método para validar o ano. Ele não deve ser negativo. Caso contrário inicializar o atributo com 2023;
  - Crie métodos para atribuir os valores aos atributos – um método para cada atributo, não esqueça de validar os dados;

# PRÁTICA 2

---



- Crie métodos para retornar os valores dos atributos – um método para cada atributo;
- Crie um método que retorne o nome do mês de acordo com o número que está armazenado no atributo mês;
- Crie uma classe para testar a classe Data;
  - Leia do usuário os valores do dia, mês e ano;
  - Instancie a classe Data e informe os valores;
  - Exiba a data formatada e o nome do mês;

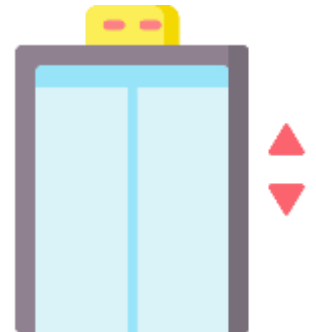


# PRÁTICA 3

---



- Crie uma **classe** denominada **Elevador** para armazenar as informações de um **elevador dentro de um prédio**. A classe deve armazenar o **andar atual** (0=térreo), **total de andares** no prédio, excluindo o térreo, **capacidade** do elevador (em pessoas) e **quantas pessoas** estão presentes nele;
  - A classe deve também disponibilizar os seguintes **métodos**:
  - **inicializa**: deve receber como parâmetros: a capacidade do elevador e o total de andares no prédio;
  - **entra**: acrescenta uma pessoa no elevador;
  - **sai**: remove uma pessoa do elevador;
  - **sobe**: sobe um andar;
  - **desce**: desce um andar;



# Copyright © 2020 - 2025

## Prof. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*“Nosso maior medo não deve ser o fracasso, mas ser bem-sucedidos em algo que não importa”*