FIAP GRADUAÇÃO

FIAP

DOMAIN DRIVEN DESIGN

Prof. Me. Thiago T. I. Yamamoto

#05 – ENCAPSULAMENTO

 *profthiagoy@fiap.com.br*  *thiagoyama*

TRAJETÓRIA

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

#05 - AGENDA

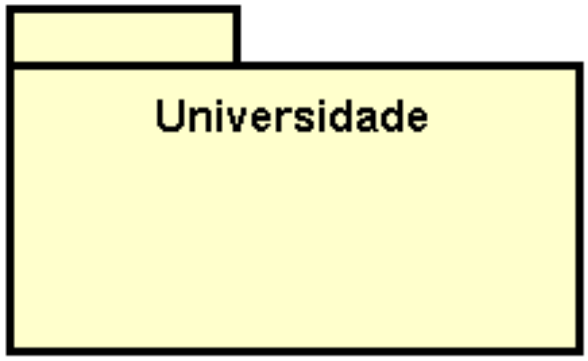
- Pacotes
- Modificadores de Acessos
- Encapsulamento
- Java Beans

PACOTES

PACOTES

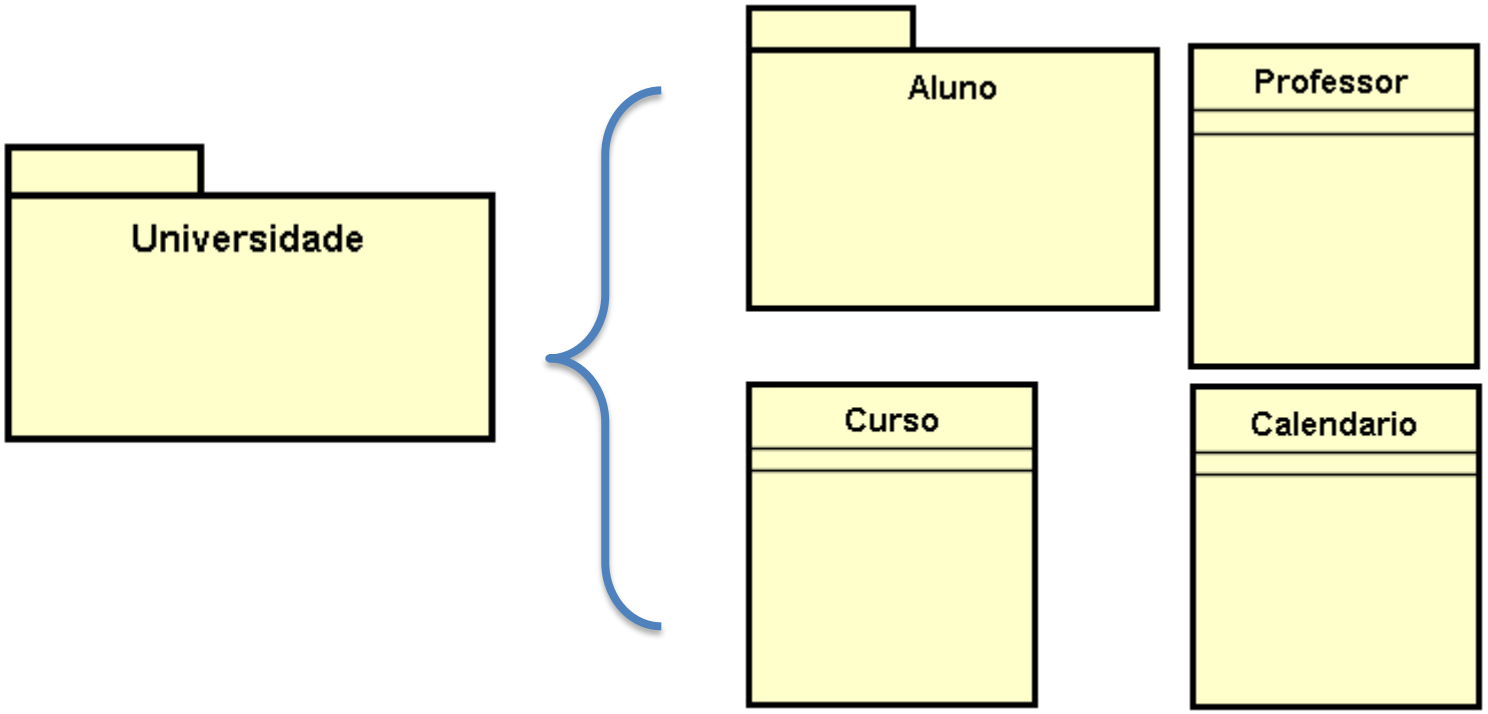
- Um mecanismo de propósito geral para **organizar elementos em grupos;**
- Um **elemento do modelo** que pode conter **outros elementos** do modelo;
- O **pacote** pode ser usado:
  - Para **organizar** o modelo em desenvolvimento;
  - Como uma **unidade de gerenciamento de configuração**;

Representação **Gráfica**:



PACOTES

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



PACOTES

FIAP

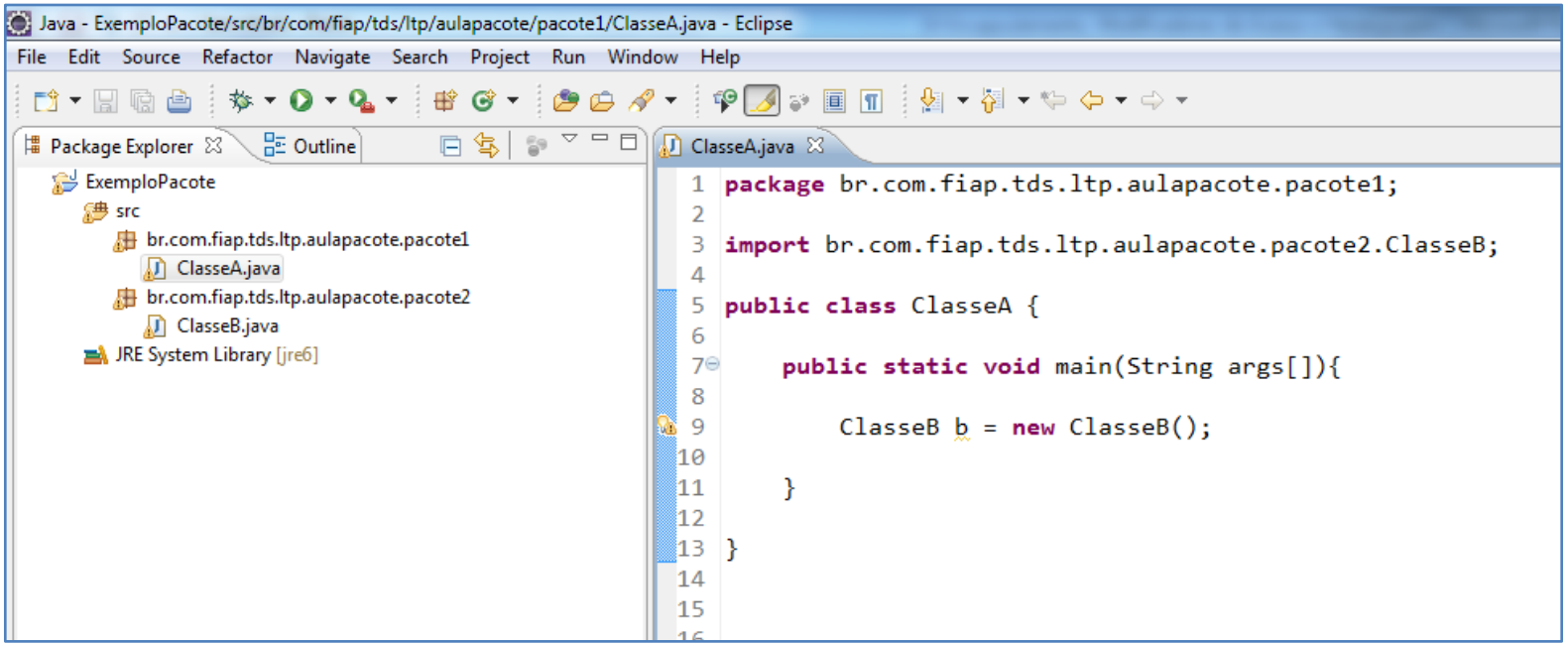
- Além das **classes**, o **Java** provê um recurso adicional que ajuda a modularidade: **o uso de pacotes;**
- **Pacotes** permitem a criação de **espaços de nomes**, além de mecanismos de controle de acesso;
- **Pacotes** são tipicamente implementados como **diretórios**;
- Os **arquivos das classes**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

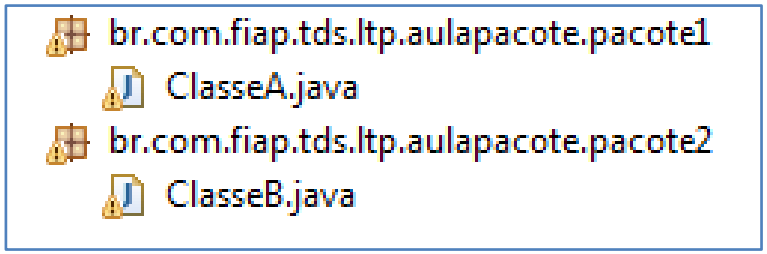
PACOTES

- "**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**;

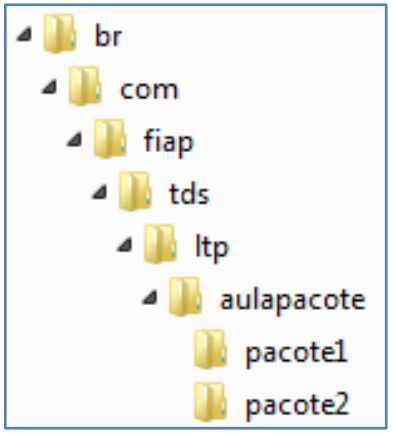
- **Importação** de Pacotes:
  - Podemos usar o **nome simples** (não qualificado) de uma classe que pertença a um pacote se **importarmos a classe**;
  - A importação de uma classe (ou classes de um pacote) pode ser feita no **início do arquivo**, **após a declaração do pacote** (se houver);
  - As classes do pacote padrão **java.lang não precisam ser importadas** (Ex.: String)

PACOTES

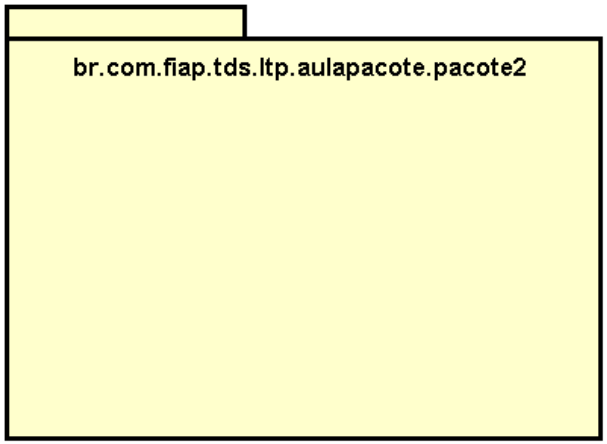

Java - ExemploPacote/src/br/com/fiap/tds/ltp/aulapacote/pacote1/ClasseA.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Outline
ExemploPacote
src
br.com.fiap.tds.ltp.aulapacote.pacote1
ClasseA.java
br.com.fiap.tds.ltp.aulapacote.pacote2
ClasseB.java
JRE System Library [jre6]
ClasseA.java
1 package br.com.fiap.tds.ltp.aulapacote.pacote1;
2
3 import br.com.fiap.tds.ltp.aulapacote.pacote2.ClasseB;
4
5 public class ClasseA {
6
7 public static void main(String args[]){
8
9 ClasseB b = new ClasseB();
10
11 }
12
13 }
14
15


br.com.fiap.tds.ltp.aulapacote.pacote1
ClasseA.java
br.com.fiap.tds.ltp.aulapacote.pacote2
ClasseB.java

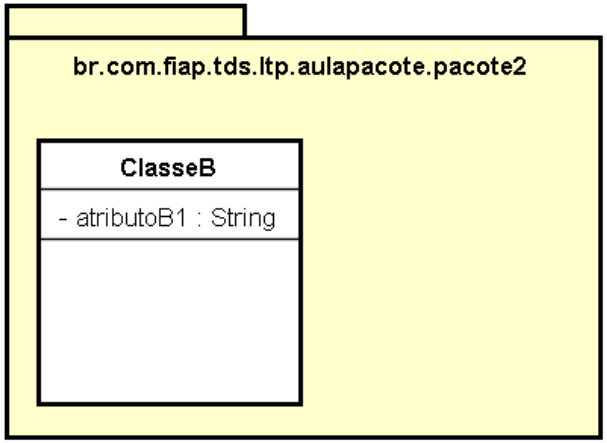
=


br
com
fiap
tds
ltp
aulapacote
pacote1
pacote2

PACOTES




```
package br.com.fiap.tds.ltp.aulapacote.pacote1;
```




```
package br.com.fiap.tds.ltp.aulapacote.pacote2;
```

MODIFICADORES DE ACESSO

MODIFICADORES DE ACESSO

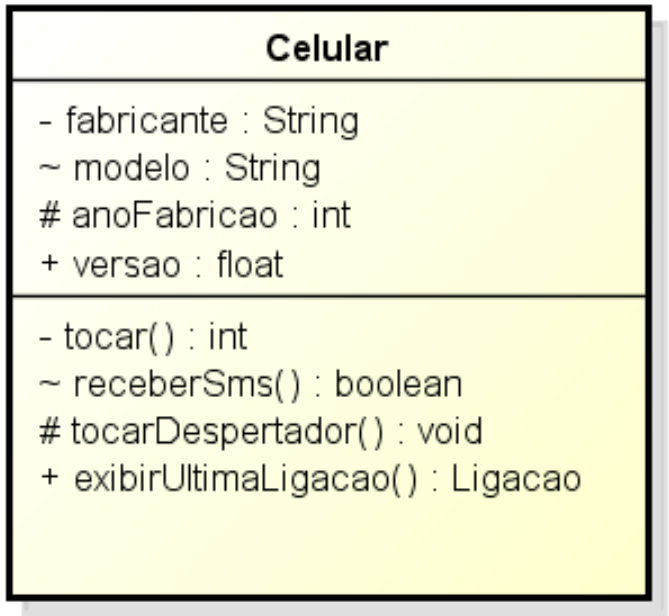
- **As linguagens OO** disponibilizam formas de **controlar o acesso** aos membros (atributos e métodos) de uma classe. No mínimo, podemos controlar o que é **público** ou **privado**;
- O **Java** disponibiliza três modificadores de acesso:
  - private
  - protected
  - public
- **Quando nenhum modificador** 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;

MODIFICADORES DE ACESSO

Símbolo	Palavra-chave	Descrição
-	private	Atributos e métodos são acessíveis somente nos métodos da própria classe. Este é o nível mais rígido 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 menos rígido de encapsulamento.

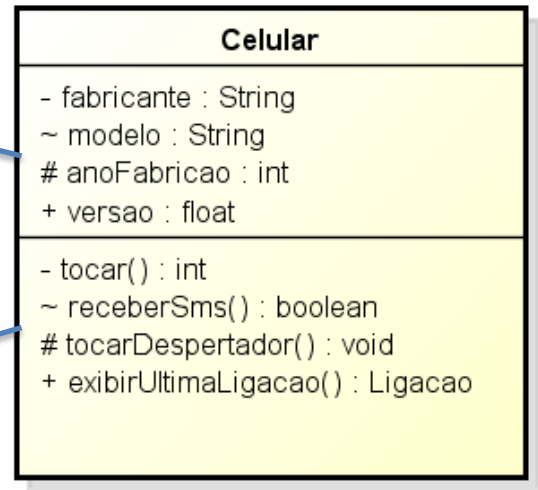
MODIFICADORES DE ACESSO

- Os **modificadores de acesso** podem ser **representados** no **diagrama de classes** através dos símbolos:
  - **-** (private)
  - **~** (*default*)
  - **#** (protected)
  - **+** (public)



MODIFICADORES DE ACESSO

```
private String fabricante;
String modelo;
protected int anoFabricacao;
public float versao;


private int tocar(){ ... }
boolean receberSms(){ ... }
protected void tocarDespertador(){ ... }
public Ligacao exibirUltimaLigacao(){ ... }
```



ENCAPSULAMENTO

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;**

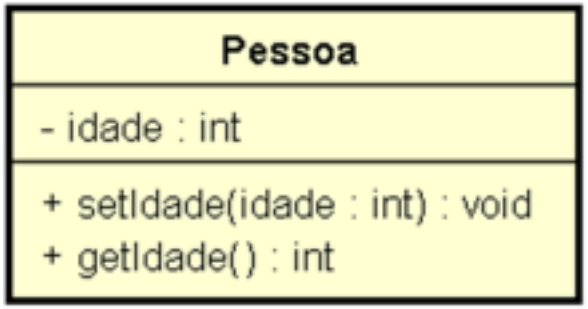
ENCAPSULAMENTO

- **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**;

ENCAPSULAMENTO - IMPLEMENTAÇÃO

- 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;
        }

}
```



ENCAPSULAMENTO - IMPLEMENTAÇÃO

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

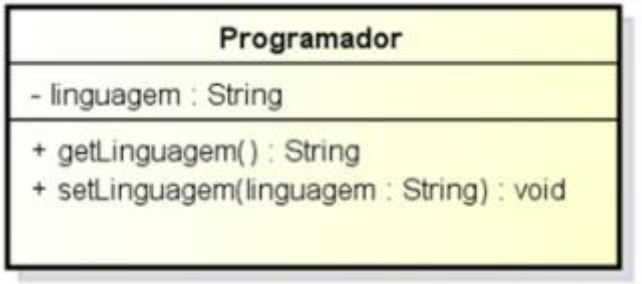
```
public class Telefone{

        private String ddd, numero;

        public String getTelefoneFormatado(){
          return formatarTel(ddd,numero);
        }

     private String formatarTel(String ddd, String numero){
          return “(” + ddd + ”) ” + numero;
        }

}
```

JAVA BEANS

- 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**;

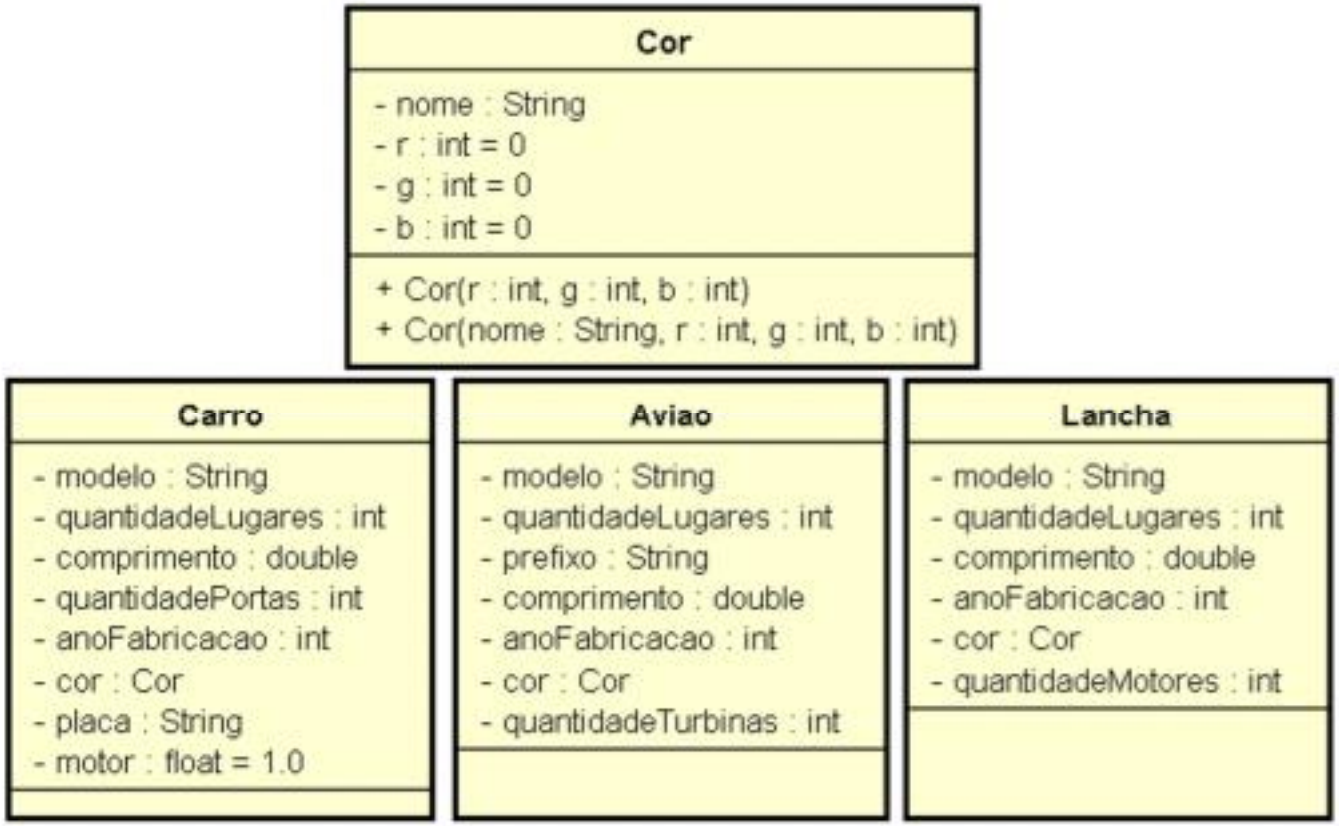
JAVA BEANS

```
public class Programador {
        private String linguagem;

        public String getLinguagem(){
                return linguagem;
        }

        public void setLinguagem(String linguagem){
                this.linguagem = linguagem;
        }
}
```



PRÁTICA

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



PRÁTICA

- 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 2020;
  - Crie métodos para atribuir os valores aos atributos – um método para cada atributo –, não esqueça de validar os dados;

PRÁTICA

- 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;

FIAP

Copyright © 2020 - 2022
Prof. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proíbido 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"