DOMAIN DRIVEN DESIGN

Prof. Rafael Desiderio

05 – Encapsulamento

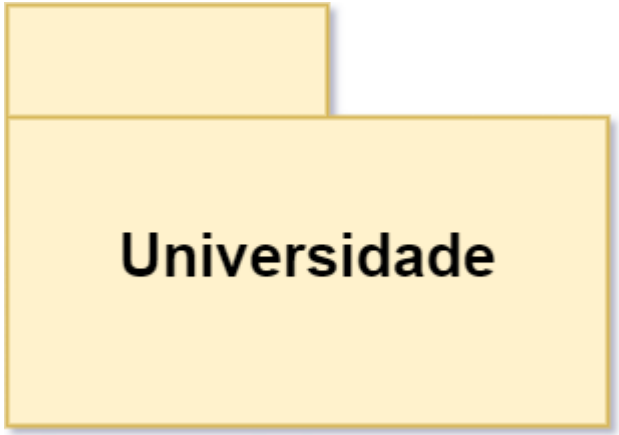
FIAP

Pacotes

Pacotes

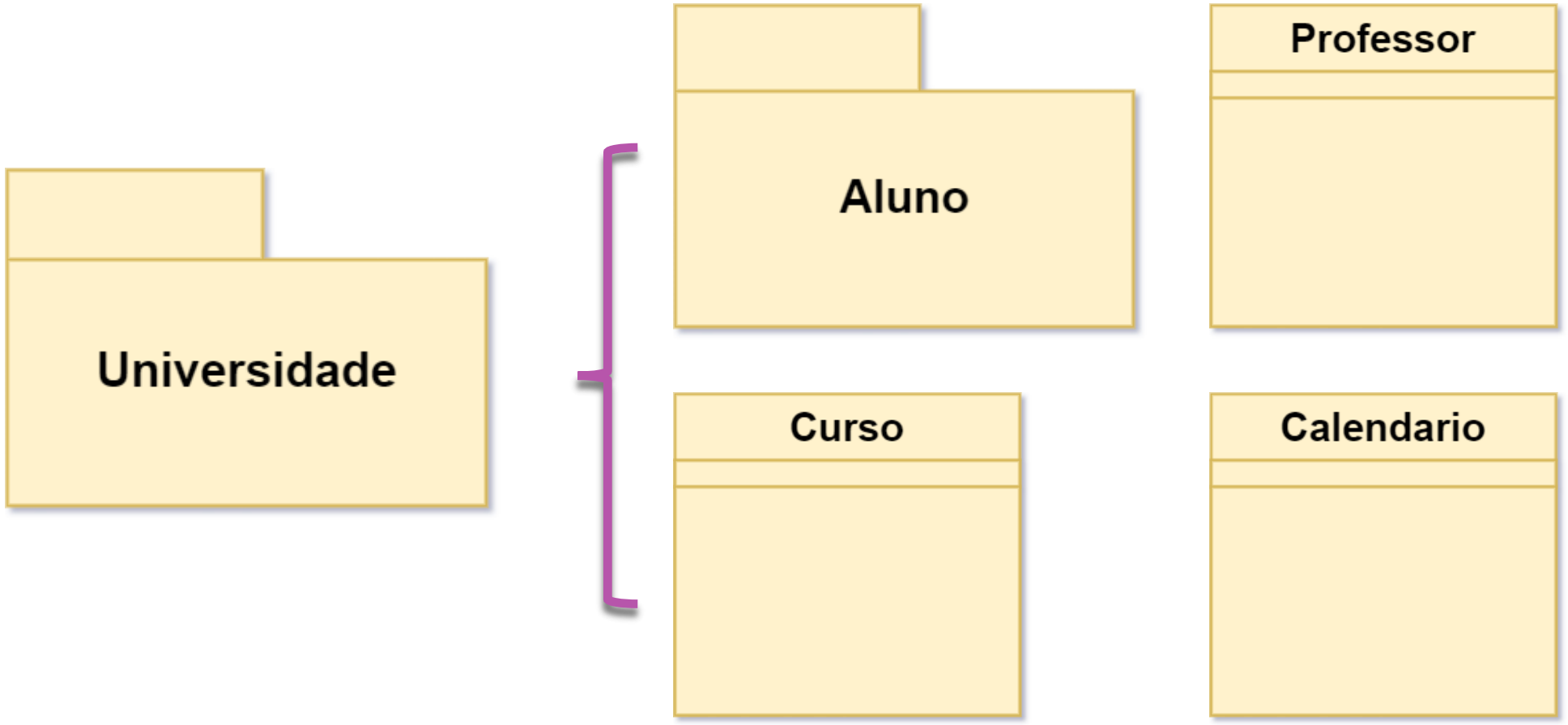
- Um mecanismo de propósito geral para **organizar elementos em grupos;**
- Um **elemento do modelo** que pode conter **outros elementos** do modelo;
- O **pacote** pode ser usado:
  - Para **organizar** o modelo em desenvolvimento;
  - Como uma **unidade de gerenciamento de configuração**;

Representação **Gráfica**:



Pacotes

- Um **pacote** pode conter **classes** e **outros pacotes**:
  - Nesse exemplo, o pacote **Universidade** contém um pacote **Aluno** e três classes:



Pacotes

- 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

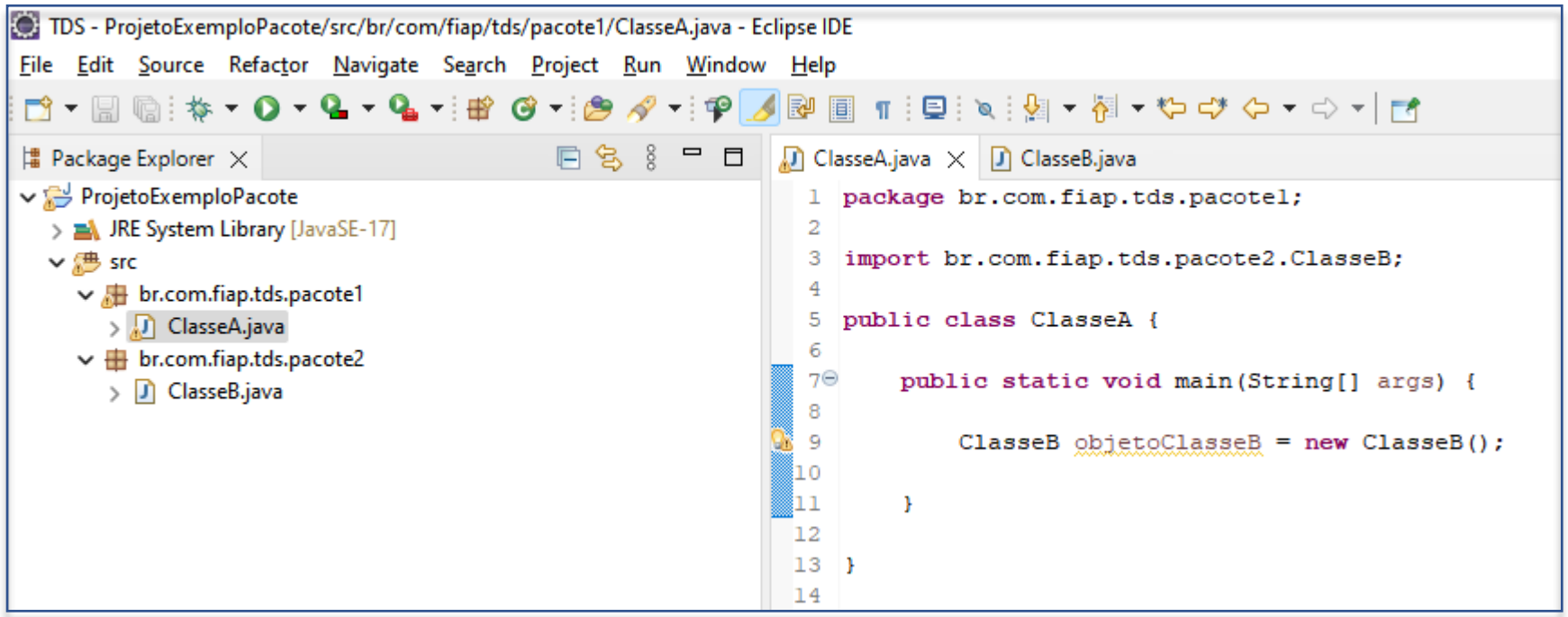
- "**Empacotando**" uma **Classe**:

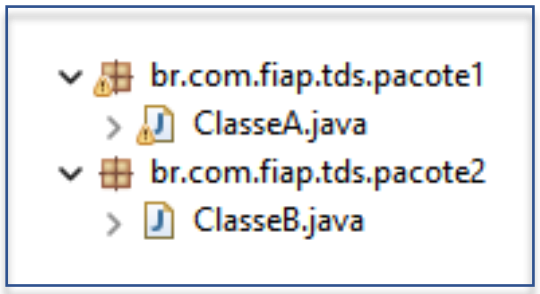
  Para declararmos uma **classe** como pertencente a um **pacote**, devemos:

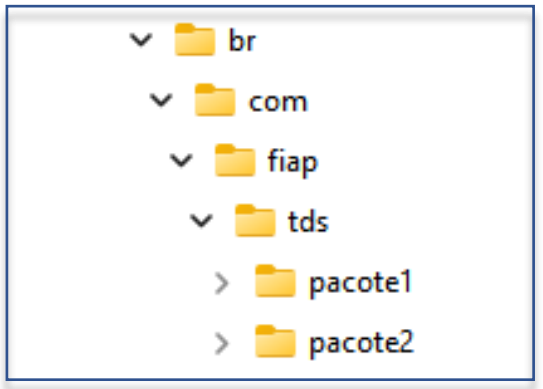
  - declará-la em um **arquivo dentro do diretório** que representa o pacote;
  - declarar, na **primeira linha do arquivo**, que a **classe** pertence ao **pacote**;

- **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);

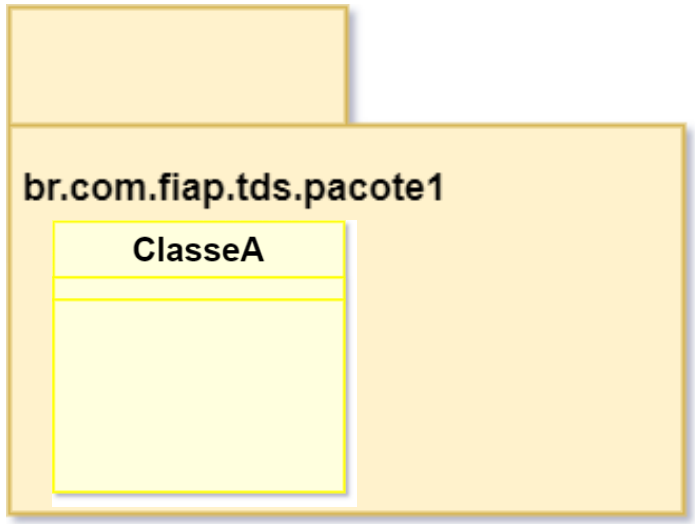
Pacotes


TDS - ProjetoExemploPacote/src/br/com/fiap/tds/pacote1/ClasseA.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ProjetoExemploPacote
JRE System Library [JavaSE-17]
src
br.com.fiap.tds.pacote1
ClasseA.java
br.com.fiap.tds.pacote2
ClasseB.java
ClasseA.java
ClasseB.java
1 package br.com.fiap.tds.pacote1;
2
3 import br.com.fiap.tds.pacote2.ClasseB;
4
5 public class ClasseA {
6
7 public static void main(String[] args) {
8
9 ClasseB objetoClasseB = new ClasseB();
10
11 }
12
13 }
14

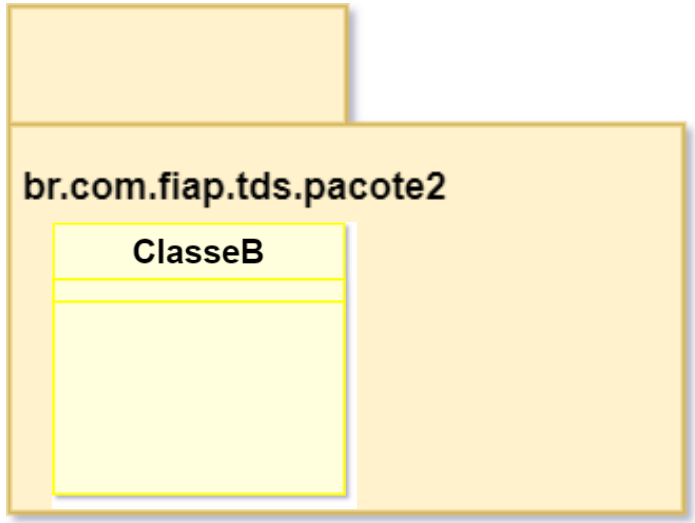

br.com.fiap.tds.pacote1
ClasseA.java
br.com.fiap.tds.pacote2
ClasseB.java


br
com
fiap
tds
pacote1
pacote2

Pacotes




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




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

FIAP

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

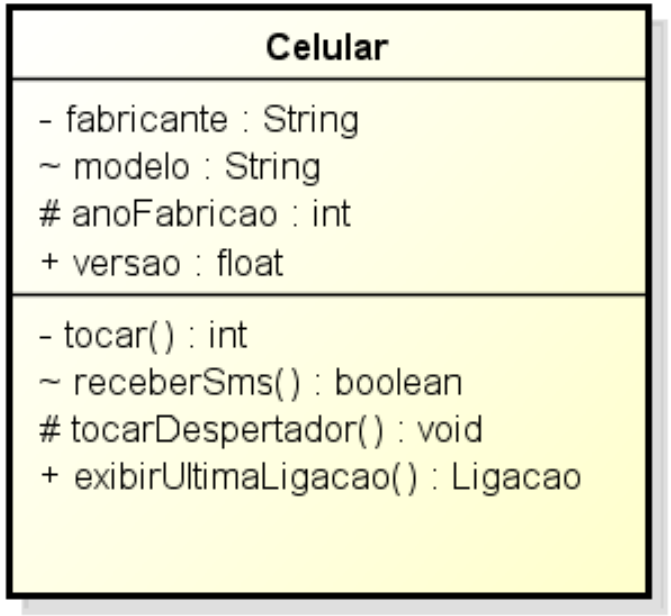
- Os **modificadores de acesso** podem ser **representados** no **diagrama de classes** através dos símbolos:

- (private)

~ (*default*)

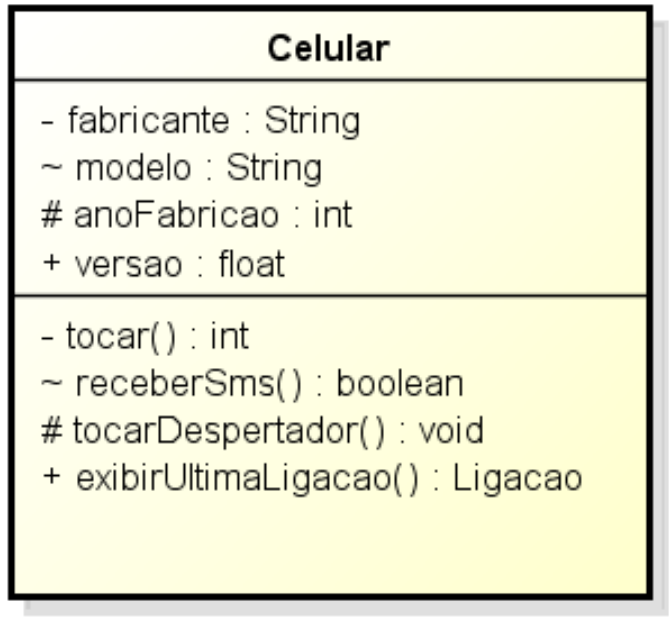
 # (protected)

+ (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(){ ... }
```



FIAP

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;**

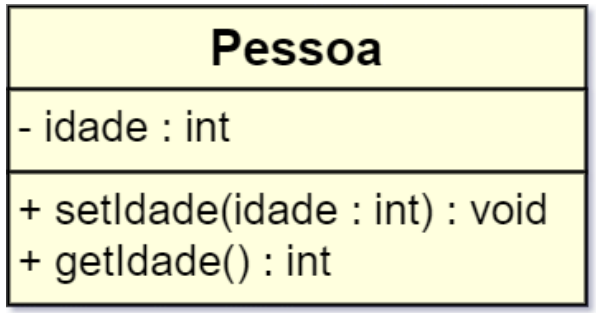
Encapsulamento

- **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:

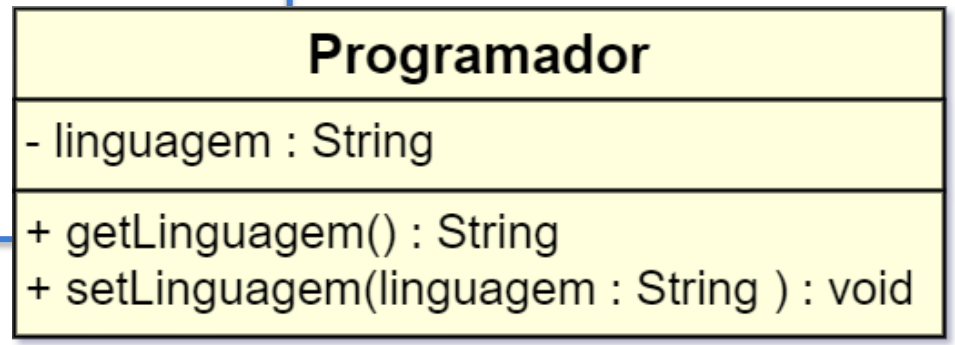
```
public class Telefone{

        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**;

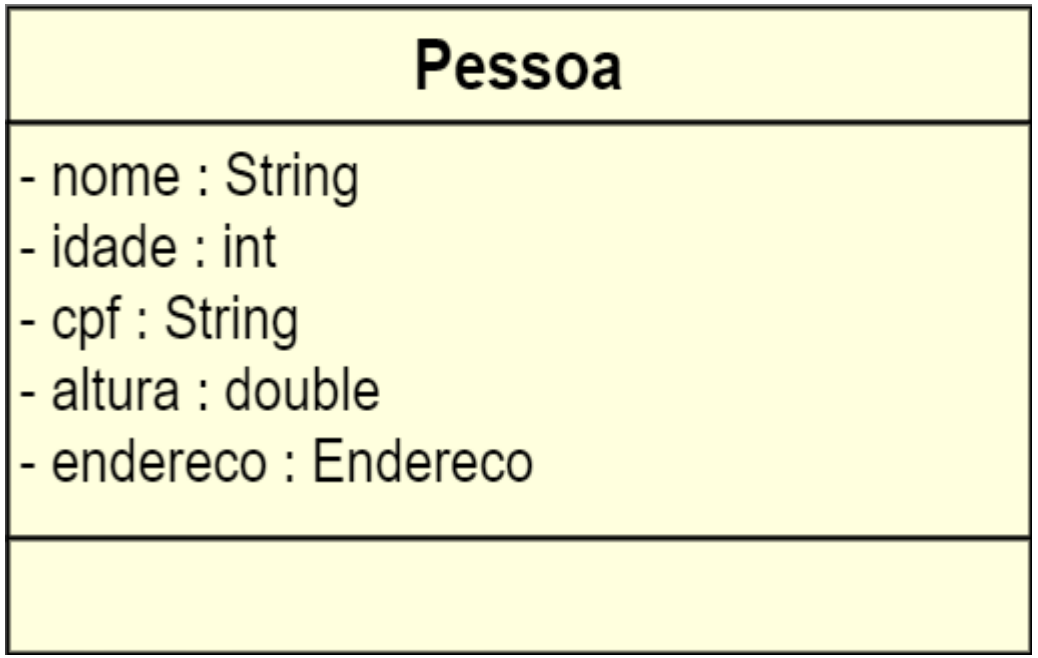
Java Beans

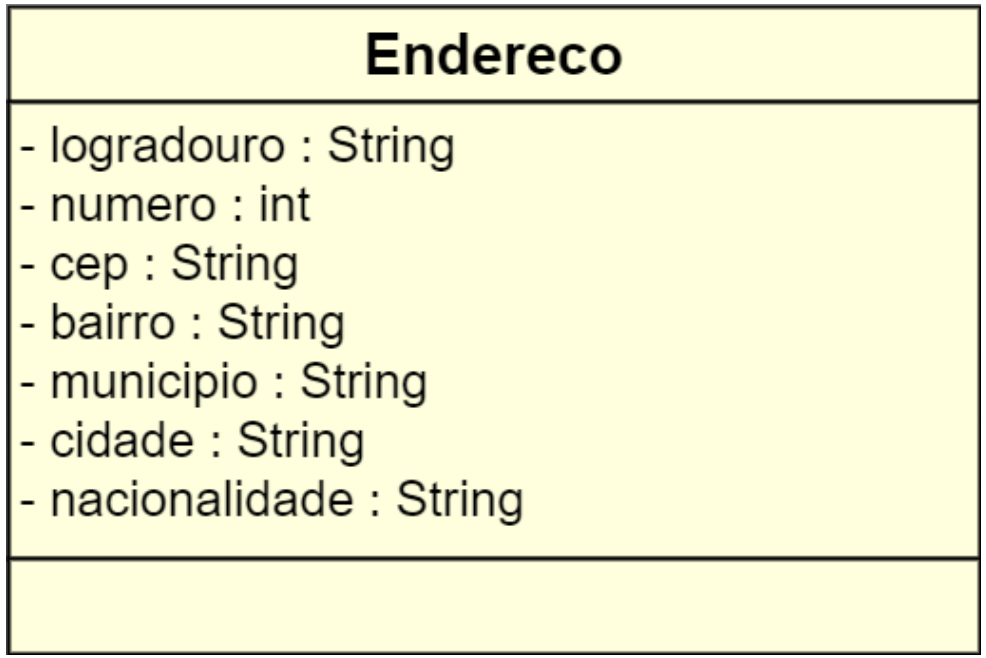
```
public class Programador {
        private String linguagem;

        public String getLinguagem(){
                return linguagem;
        }

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



Vamos à prática

- Crie as **classes** abaixo e inclua os métodos gets e sets;
- **Instancie** as **classes** em uma classe de teste (ex: TesteMain), adicione os valores para exibir;





FIAP

"Somos o que repetidamente fazemos. Portanto, a excelência não é um feito, é um hábito". - Aristóteles