PROGRAMAÇÃO ORIENTADA A OBJETOS

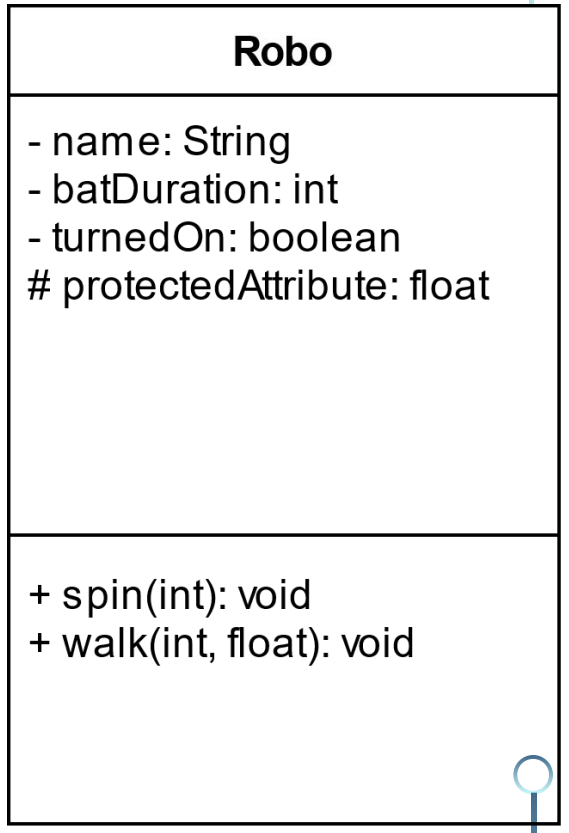
PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/poo

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

ENCAPSULAMENTO

- Pilar do paradigma POO usualmente associado ao fato de **ESCONDER/OCULTAR** os detalhes de implementação dos métodos em determinada Classe
- Tem forte conexão com a realidade, onde os dispositivos que usamos são acessíveis por meio de INTERFACES públicas abertas a interações com outros objetos, e não nos preocupamos no COMO são organizados, construídos etc., mas temos acesso às funcionalidades.
  - Exemplos: SMARTPHONE (touchscreen com "teclas", câmera, microfone); CARRO (volante, ignição, pedais, câmbio...); TERMINAL BANCÁRIO (touchscreen, teclas/botões, leitor cartão)



ENCAPSULAMENTO

- Remete a ter algo com visibilidade PRIVADA (-) aos métodos da própria classe onde é declarado
- Um método PÚBLICO (+) "exportaria" sua interface (assinatura) às outras classes, independente de estarem na mesma pasta ou em outras pastas (pacotes/packages).
- Basta saber usar/chamar o método (seu nome, quantidade e tipo de parâmetros e tipo de retorno), não importa sua implementação às demais classes. O modificador de acesso + se aplica a classes, métodos, atributos, construtores

ENCAPSULAMENTO

- Os atributos PRIVADOS de uma classeA podem ser acessados por outras classes (B, C, etc.) através dos métodos PÚBLICOS get (leitura) e set (escrita) implementados na classeA, também chamados GETTERS e SETTERS, que podem ser gerados automaticamente nas IDEs atuais.
- O get retorna valor (o atributo sendo consultado)
- O set recebe parâmetro, atribui ao atributo da classe e não há retorno (void)

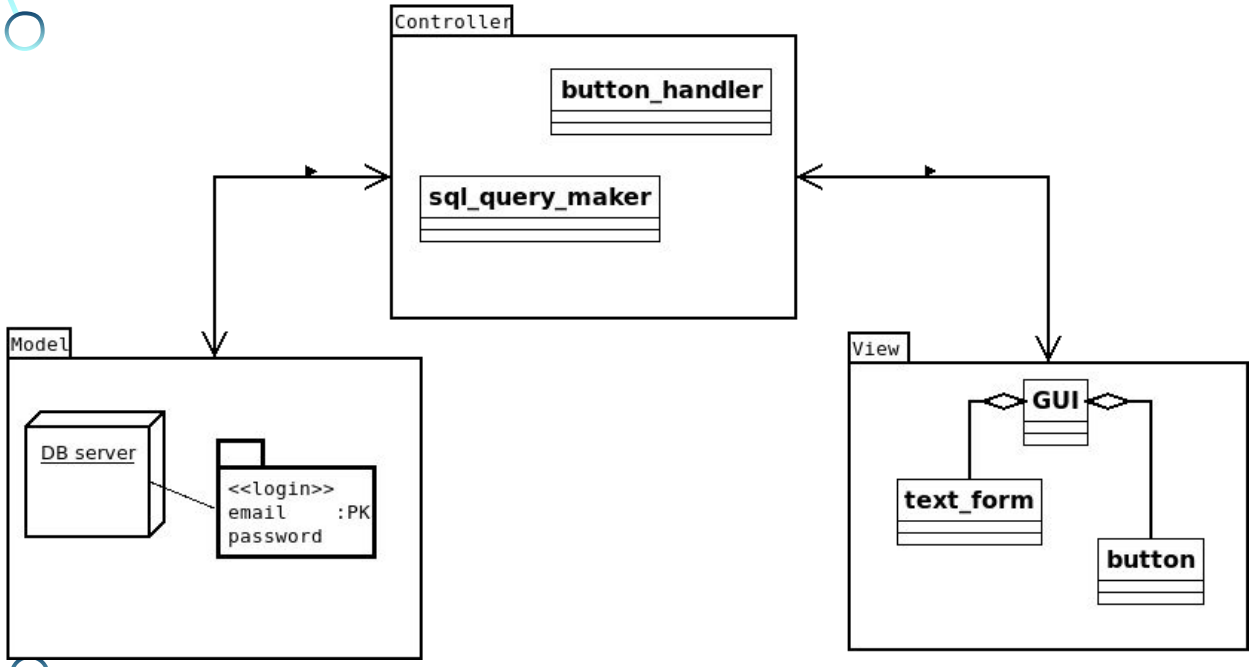
public Y getX() retorna atributo do tipo Y

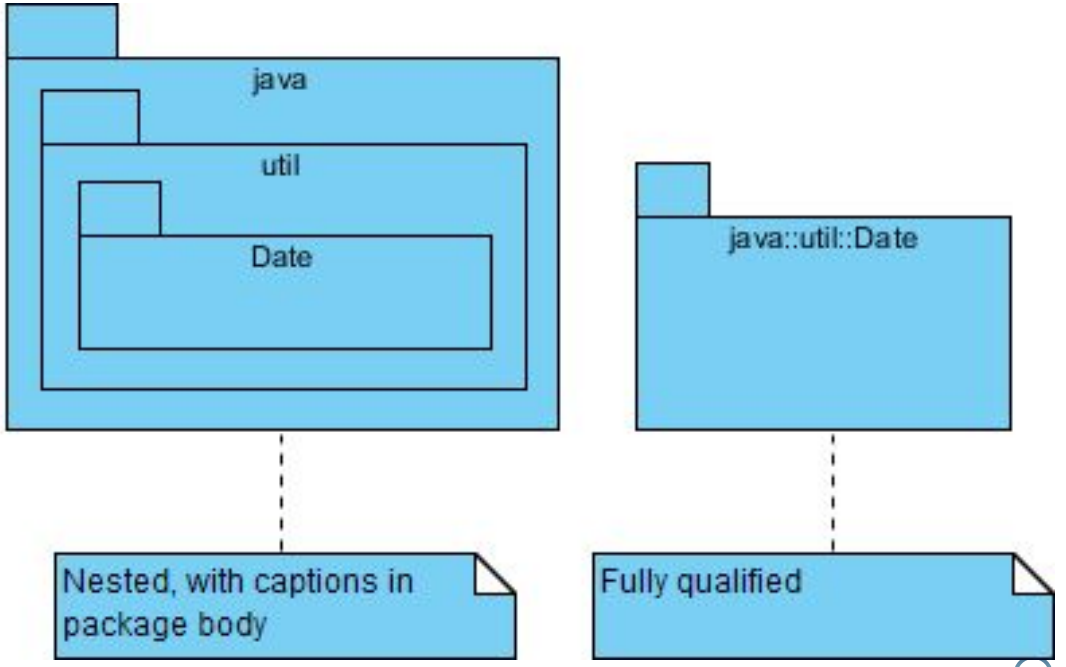
```
public String getNome () {
    return nome;
}

public int getIdade() {
    return idade;
}
```

O this (cláusula) faz referência ao próprio objeto instanciado

```
public void setNome(String nome) {
    this.nome = nome;
}
public void setIdade(int idade) {
    this.idade = idade;
}
```

ENCAPSULAMENTO - PACOTES





PACOTES - um dos diagramas UML (exemplificando)

https://2019-2-arquitetura-desenho.github.io/wiki/dinamica_seminario_III/diagrama_pacotes/

ENCAPSULAMENTO - PACOTES

- Quando organizamos nossas classes por similaridade (entidades, serviços, modelos, UIs, acesso à bases de dados etc.) costumamos criar Pacotes (Packages) – No VSCODE é uma pasta
- A visibilidade PROTEGIDA (protected #) permite que métodos/atributos de uma **superclasse** possam ser acessados por suas **subclasses** (herança, mesmo que em outro pacote) e por outras classes no mesmo pacote
- Se nenhuma visibilidade for informada, vale o escopo de PACOTE, ou seja, as classes do mesmo PACOTE acessam métodos e atributos protegidos de outra classe. Chamada visibilidade *default* ou *friendly*.

ENCAPSULAMENTO – VISIBILIDADES (RESUMO)

Modificador	Acessível por classes no mesmo pacote	Acessível por classes em outros pacotes	Acessível por subclasses no mesmo pacote	Acessível por subclasses em outros pacotes
Público (public+)	Sim	Sim	Sim	Sim
Protegido (protected#)	Sim	Não	Sim	Sim
Default (pacote)	Sim	Não	Sim	Não
Privado (private-)	Não	Não	Não	Não