Programação Orientada a Objetos: POO

Visibilidade em Objetos

Linguagem de Modelagem Unificada
LMU > UML

Não vamos aprofundar, mas precisamos utilizar para falar de POO

Diagrama de Classes

Diagrama de Classes serve para simplificar o Molde (a Classe) ao extremo

Visibilidade em objetos

Toda Classe = Retângulo!

Não preciso desenhar uma caneta, desenho apenas um retângulo chamado Caneta

Não preciso desenhar um carro, desenho apenas um retângulo chamado Carro

Não preciso desenhar um pedido, desenho apenas um retângulo chamado Pedido

Visibilidade em objetos

Diagrama de Classes

Dividido em 3 partes

- Nome
- atributos
- métodos()

Visibilidade em objetos

Modificadores de Visibilidade

Indicam o nível de acesso aos componentes internos de uma Classe

- atributos
- métodos

São 3 as visibilidades padrões:

- pública
- privada
- protegida

A visibilidade default do Java é “público pacote”

Visibilidade em objetos

+: Público - qualquer um pode utilizar, de qualquer lugar

-: Privado - somente eu posso utilizar

#: Protegido - somente quem possuir autorização pode utilizar

Visibilidade em objetos

Para o POO:

+ : público - a classe atual e todas as outras classes

- : privado - somente a classe atual - ela mesma

: protegido - a classe atual e todas as suas subclasses (veremos em herança)

Visibilidade em objetos

Porque existe público, privado e protegido?

Público: Botões

Privado: Componentes

“Protegido: Pilhas”

Atividade avaliativa n1.2

a. Utilizando a Classe criado no exercício anterior (n1.1), crie um Diagrama de Classes

b. Determine visibilidades para os atributos e métodos da Classe

c. Determine a visibilidade dos atributos e métodos da Classe criada no exercício anterior (n1.1)

```
// Pseudo código exemplo
Classe Caneta
  publico modelo: String
  publico cor: String
  privado ponta: double
  protegido carga: double
  protegido tampada: boolean

  publico Metodo escrever()
    Se (tampada) entao
      Escreva(erro)
    senao
      Escreva("qq coisa")
  FimDoMetodo

  privado Metodo tampar()
    tampada = verdadeiro
  FimDoMetodo

  privado Metodo destampar()
    tampada = falso
  FimDoMetodo

FimClasse
// ---------------------------------------
//nome do objeto | criar | Classe
caneta01 = nova Caneta
caneta01.cor = "vermelha"
//caneta01.ponta = 0.5 // não é possivel, pois o atributo é privado
//caneta01.tampada = false // não é possivel, pois o atributo é protegido
caneta01.escrever()

//tampar() // não é possivel, pois o atributo é privado

//Observação:
        // Posso definir um atributo como privado
        // Criar um metodo publico que altera o atributo
        // Chamar o atributo para metodo publico e, atraves dele, alterar o atribu
```