Bônus: Aprenda Fundamentos

(Spring e desenvolvimento de sistemas)

Capítulo: Componentes e injeção de dependência

https://devsuperior.com.br

DISCLAIMER

Este é um **BÔNUS** concedido aos alunos do Bootcamp Spring, sem custo adicional.

O objetivo deste bônus é ensinar os seguintes fundamentos:

- Injeção de dependência
- DTO, padrão camadas
- JPA: entidades associadas, lazy loading

Vale ressaltar também que o objetivo deste bônus **NÃO É** implementar um projeto completo, mas sim abordar os fundamentos citados acima.

Bons estudos!

Sistema e componentes

- Sistema é composto de componentes
- Componentes devem ser
  - Coesos (responsabilidade clara e única)
  - Desacoplados entre si
- Objetivos
  - Flexibilidade
  - Manutenção/substituição facilitada
  - Reaproveitamento

Exemplo didático

Fazer um programa ler os dados de um funcionário (nome, salário bruto) e depois mostrar o salário líquido, que consiste no valor bruto descontado impostos e previdência.

REGRAS:

1) Imposto é 20%

2) Previdência é 10%

Exemplo:

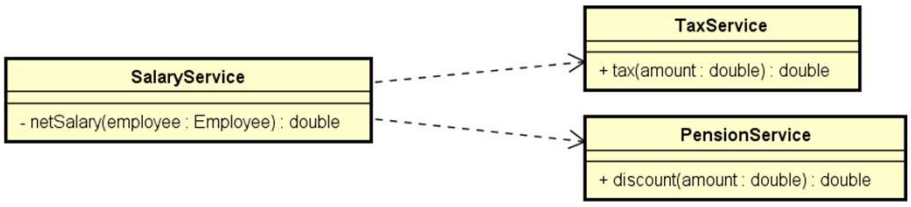
```
Nome: Maria
Salário bruto: 4000.00
Salario liquido = 2800.00
```

Solução "rápida"

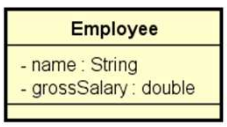
```
Locale.setDefault(Locale.US);
Scanner sc = new Scanner(System.in);

System.out.print("Nome: ");
String name = sc.nextLine();
System.out.print("Salario bruto: ");
double grossSalary = sc.nextDouble();

double netSalary = grossSalary * 0.7;
System.out.printf("Salario liquido = %.2f%n", netSalary);

sc.close();
```


Solução pensando em componentes

O sistema será composto por **componentes**, cada um com sua responsabilidade.



Modelo de domínio:



Dica: repare que temos basicamente dois tipos de objetos em nosso sistema: componentes e objetos de dados (model, entity, dto)

Inversão de controle

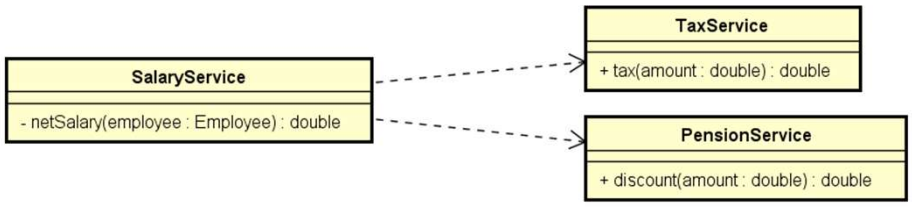
Analogia do carro:

- No carro, o motor depende da bateria
- Porém, a base de encaixe da bateria é fora do motor
- Por que não colocar a base da bateria dentro do motor?
  - Porque se for preciso trocar a bateria, não é preciso abrir o motor.

Generalizando:

- Se um componente A depende de B, A **não** deve ter o controle sobre esta dependência (B).
- Por que? Porque se for preciso trocar a dependência (B), seria preciso também abrir o componente A.
- É preciso "inverter o controle", ou seja, o controle da dependência tem que ficar fora do componente A.

Se um componente A depende de B, A **não** deve ter o controle de quem será essa dependência.




```
public class SalaryService {

    private TaxService taxService = new TaxService();
    private PensionService pensionService = new PensionService();
```

Errado

Injeção de dependência

- Uma vez que um sistema usa o princípio da inversão de controle, quando um componente A depende do componente B, esta dependência (B) precisa ser "injetada" em A.
- Em sistemas de software, essa "injeção de dependência" pode ser feita de várias formas:
  - Construtor
  - Método set
  - Container de injeção de dependência (framework)

Framework

- Tradução ao pé da letra: estrutura, armação, estrutura de trabalho
- É um conjunto de ferramentas que oferece uma infraestrutura para se desenvolver sistemas de forma produtiva
- Um framework gerencia:
  - Injeção de dependências
  - Transações (back end)
  - Ciclo de vida, escopo e reaproveitamento de componentes
  - Configurações
  - Integrações
  - Outros

Dois passos para trabalhar com injeção de dependência em frameworks

1) Registrar os componentes
2) Indicar quais componentes dependem de quais

Uma vez feito isso, o próprio framework se encarregará de:

- Instanciar componentes
- Resolver dependências
- Reaproveitar componentes
- Gerenciar escopo e ciclo de vida dos componentes