Domain Driven Design - Java

Prof. Gilberto Alexandre das Neves
profgilberto.neves@fiap.com.br

FIAP

Pacotes

Pacotes

Os **pacotes** em Java são fundamentais para organizar e estruturar um projeto de forma eficiente. Eles ajudam a separar classes relacionadas, facilitando a manutenção, a reutilização de código e a segurança.

Organização do Código

- Em projetos grandes, manter todas as classes em um único diretório se torna inviável.
- Pacotes permitem agrupar classes por funcionalidade, como **model**, **controller**, **service**, etc.

Evita Conflito de Nomes

- Se houver duas classes com o mesmo nome em um projeto (ex.: **Cliente**), elas podem coexistir em pacotes diferentes, como com.empresa.vendas.Cliente e com.empresa.suporte.Cliente.

Pacotes

FIAP

Facilita o Reuso de Código

- Classes organizadas em pacotes podem ser reutilizadas em diferentes partes do projeto ou até em outros projetos, sem precisar refatorar código.

Controle de Acesso (Encapsulamento)

- Pacotes ajudam a controlar a visibilidade de classes e métodos.
- O modificador **protected** permite acesso apenas dentro do mesmo pacote ou subclasses.
- O modificador padrão (sem especificar **public** ou **private**) restringe o acesso ao próprio pacote.

Facilidade na Manutenção e Escalabilidade

- Se o projeto crescer, novas funcionalidades podem ser adicionadas em novos pacotes sem bagunçar o código existente.

Pacotes

FIAP

Evitar Conflitos de Nomes

Se cada empresa ou desenvolvedor seguisse um padrão comum (por exemplo, começando pelo nome do projeto), haveria grandes chances de colisão de nomes. Ao usar um domínio da web ao contrário, garante-se que cada pacote seja único, pois domínios são exclusivos.

Exemplo:

- Se duas empresas diferentes criassem um pacote **utilidades.arquivo**, haveria conflitos.
- Mas se cada empresa usasse o nome do domínio invertido, os pacotes ficariam distintos:
  - **com.google.utilidades.arquivo**
  - **org.apache.utilidades.arquivo**

Isso evita sobreposição quando diferentes bibliotecas ou projetos são usados no mesmo ambiente.

FIAP

JavaBean

JavaBean

Em Java um componente, denominado de **JavaBean** ou **bean**, é um objeto que segue a especificação **JavaBean**. Para ser considerada como um **JavaBean**, uma classe precisa seguir algumas convenções de nomenclatura de métodos, construtores e comportamento.

De forma resumida, os **JavaBeans** têm as seguintes características:

- Possuem um **construtor** sem argumentos (vazio);
- Definem **métodos get/set** para acesso às suas propriedades;

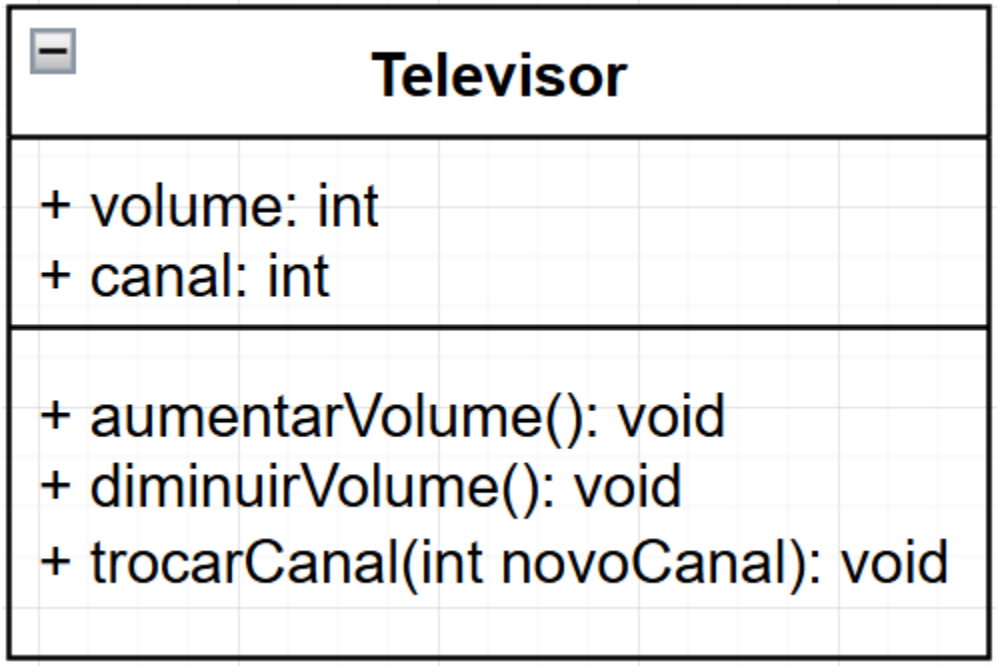
As classes **JavaBean** devem ter nenhuma (ou o mínimo possível) de implementação das **regras de negócios** do projeto.

Para melhor organização do projeto as classes **JavaBean** são organizadas dentro de um pacote próprio (por exemplo, **bean**) e as classes de teste com o método **main** dentro de outro pacote (por exemplo, **main**).

Praticando…

Praticando

Implemente a classe abaixo:



Agora crie uma nova classe (~~**`UsaTelevisor`**~~ **`Main`**) com o método **`main`** para criar objetos e testar objetos da classe **`Televisor`**. Ao final exiba o volume e canal atuais.

A classe Televisor

```java
package br.com.fiap.bean;

public class Televisor {
    // atributos
    public int volume;
    public int canal;

    // métodos;
    public void aumentarVolume() {
        volume++;
    }
    public void diminuirVolume() {
        volume--;
    }
    public void trocarCanal(int novoCanal) {
        canal = novoCanal;
    }
}
```

Métodos printf() e String.format()

Método printf()

O método **printf** em Java é usado para **formatar e exibir saídas no console** de maneira mais flexível do que **System.out.println**.

Sintaxe:

```
System.out.printf(String, argumentos);
```

Principais Marcadores de Formato

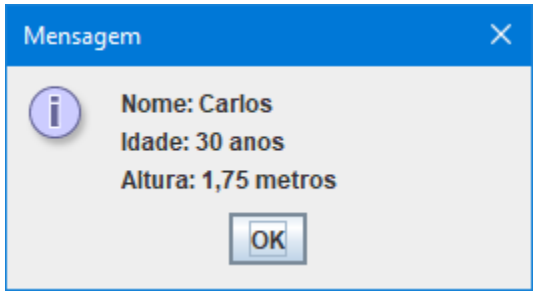
Marcador	Descrição	Exemplo
`%d`	Inteiro decimal	`System.out.printf("Valor: %d", 10);` → `Valor: 10`
`%f`	Número de ponto flutuante	`System.out.printf("Valor: %.2f", 3.1415);` → `Valor: 3.14`
`%s`	String	`System.out.printf("Nome: %s", "João");` → `Nome: João`
`%c`	Caractere	`System.out.printf("Letra: %c", 'A');` → `Letra: A`
`%n`	Quebra de linha (igual `\n`)	`System.out.printf("Linha 1%nLinha 2");` → `Linha 1` (nova linha) `Linha 2`

Método String.format()

O método **String.format()** funciona como **printf**, mas retorna uma **String formatada** que pode ser usada por exemplo com a classe **JOptionPane**.

Exemplo:

```
5  public class Main {
6      public static void main(String[] args) {
7          String nome = "Carlos";
8          int idade = 30;
9          double altura = 1.75;
10
11         // Criando mensagem formatada
12         String mensagem = String.format("Nome: %s\nIdade: %d anos\nAltura: %.2f metros",
            nome, idade, altura);
13
14         // Exibindo na janela
15         JOptionPane.showMessageDialog(null, mensagem);
16     }
17 }
```

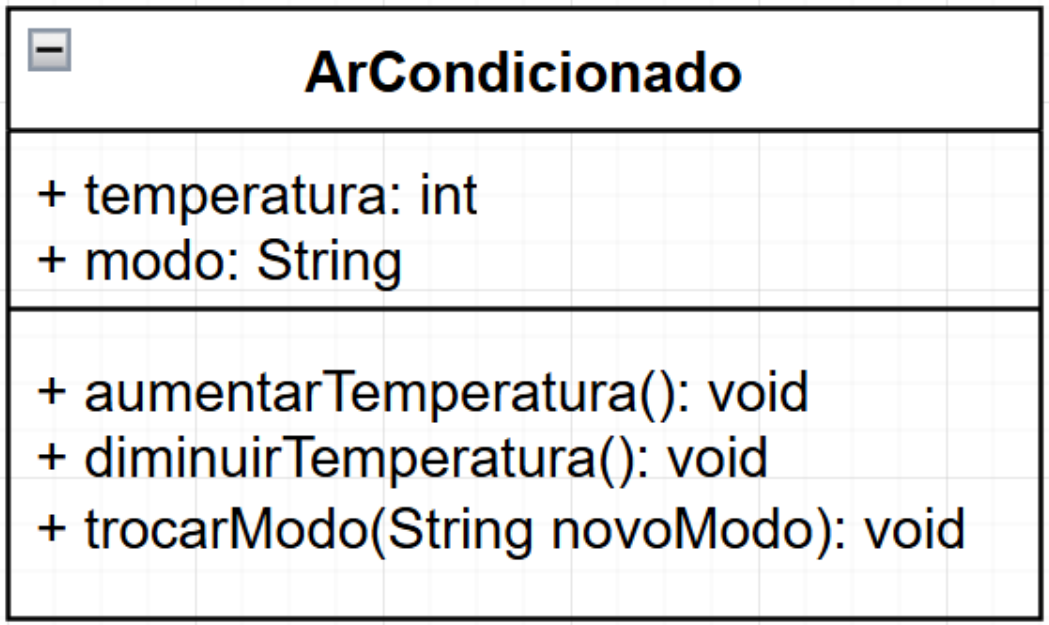


A classe Main (~~UsaTelevisor~~)

```
1   package br.com.fiap.main;
2
3   import br.com.fiap.bean.Televisor;
4
5   public class Main {
6       public static void main(String[] args) {
7           Televisor televisor = new Televisor();
8           televisor.canal = 5;
9           televisor.volume = 7;
10          televisor.trocarCanal(4);
11          televisor.diminuirVolume();
12          televisor.diminuirVolume();
13          System.out.printf("Volume atual: %d\nCanal atual: %d", televisor.volume,
             televisor.canal);
14      }
15  }
```

Praticando 2

Implemente a classe abaixo:



Agora crie uma nova classe (~~**`UsaArCondicionado`**~~ **`Main`**) com o método **`main`** para criar objetos e testar a classe **`ArCondicionado`**. Ao final exiba a temperatura e o modo atuais.

Como valores para o atributo modo, utilize: “resfriar”, “aquecer” ou “ventilar”.

A classe ArCondicionado

```java
package br.com.fiap.bean;

public class ArCondicionado {
    // atributos
    public int temperatura;
    public String modo;

    // métodos
    public void aumentarTemperatura() {
        temperatura++;
    }
    public void diminuirTemperatura() {
        temperatura--;
    }
    public void trocarModo(String novoModo) {
        modo = novoModo;
    }
}
```

A classe Main (~~UsaArCondicionado~~)

```
15
16          ArCondicionado arCondicionado = new ArCondicionado();
17          arCondicionado.temperatura = 25;
18          arCondicionado.modo = "ventilar";
19          arCondicionado.trocarModo("resfriar");
20          arCondicionado.diminuirTemperatura();
21          arCondicionado.diminuirTemperatura();
22          String mensagem = String.format("\nTemperatura atual: %d°C\nModo atual: %s",
              arCondicionado.temperatura, arCondicionado.modo);
23          System.out.println(mensagem);
24      }
25  }
```

Praticando 3

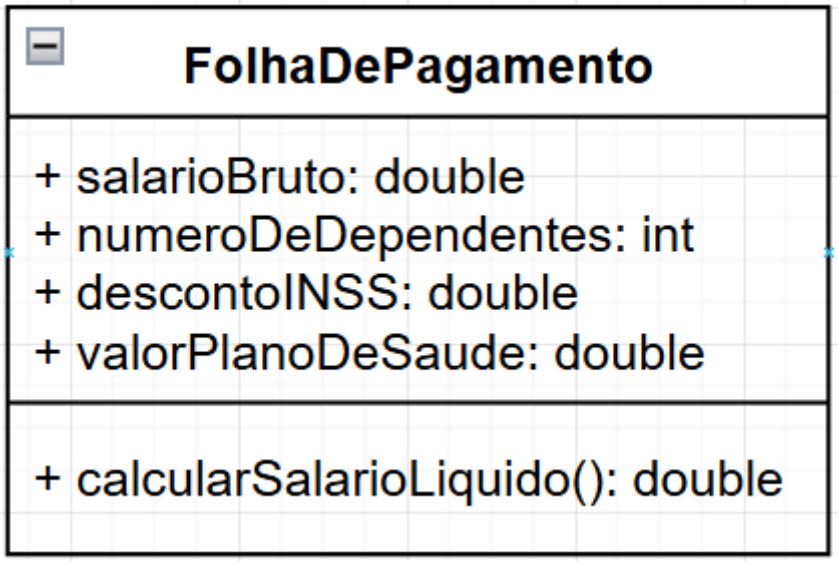
Use a criatividade e represente no modelo UML uma classe a sua escolha, defina para essa classe no mínimo 3 atributos e no mínimo 2 métodos.

Agora implemente o código desta classe no Java.

Em seguida, crie outra classe para criar um objeto da classe implementada anteriormente e modifique os valores de seus atributos, utilize seus métodos e exiba resultados.

Praticando 4

Implemente a classe abaixo:



- O método **calcularSalarioLiquido()** deve calcular o desconto de **INSS** e do **plano de saúde** (depende do número de dependentes) e retornar o valor do salário líquido (salário com todos os descontos subtraídos).
- Agora crie uma nova classe com o método **`main`** para criar objetos e testar a classe **`FolhaDePagamento`**.

Referências

Java como programar. Paul Deitel e Harvey Deitel. Pearson, 2011.

Java 8 – Ensino Didático : Desenvolvimento e Implementação de Aplicações. Sérgio Furgeri. Editora Érica, 2015.

Até breve!