ENUM

O que são, como e quando usá-los

O que é um Enum?

Enum (ou Enumeração) é um tipo de dado que consiste em um conjunto de valores constantes.

Esses valores são usados para representar estados ou categorias fixas dentro de um programa.

Dias da semana: `Segunda-feira`, `Terça-feira`, etc.

Status de um pedido: `Em andamento`, `Concluído`, `Cancelado`.

Vantagens de Usar Enums

- **Facilidade de leitura**: Ao usar Enums, o código fica mais legível, pois substituímos números mágicos ou strings por nomes mais significativos.
- **Segurança de tipo**: Ajudam a evitar erros, como usar valores inválidos ou incorretos.
- **Organização**: Tornam o código mais organizado e fácil de manter.

Sintaxe Básica (Exemplo em Java)

```
public enum DiaSemana {
    SEGUNDA, TERÇA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO
}
```

Por padrão, os valores de um enum têm índices numéricos começando de 0, mas podem ser atribuídos valores personalizados.

Pergunta para os Alunos:

Como você pode usar um enum para representar os diferentes status de um pedido em um sistema de e-commerce?

a) Apenas com os nomes dos status (como PENDENTE, EM_ANDAMENTO, etc.)
b) Associando cada status a um número único, como PENDENTE(1), EM_ANDAMENTO(2), etc.
c) Usando valores booleanos, como true ou false, para representar o status
d) Nenhuma das alternativas anteriores

Resposta correta:

b) Associando cada status a um número único, como PENDENTE(1), EM_ANDAMENTO(2), etc.

Explicação:

A alternativa **b** é a mais indicada porque associar **valores específicos** (como números) a cada status do pedido torna o código mais organizado e flexível. Isso traz várias vantagens, como:

Facilidade de armazenamento: Quando armazenamos o status em um banco de dados, números são mais eficientes e fáceis de processar.
Segurança de tipo: Ao usar valores numéricos ou outros tipos de dados associados aos enums, evitamos erros e tornamos o código mais seguro, pois só podemos usar os valores válidos definidos no enum.
Clareza e manutenção: Associar números a cada status ajuda a manter o código claro e facilita alterações no futuro, caso seja necessário modificar ou adicionar novos estados.

Por outro lado, a **alternativa a** também é válida em alguns cenários, mas o uso de valores associados permite um controle mais rigoroso e flexível, além de facilitar a integração com outros sistemas, como bancos de dados ou APIs.

A **alternativa c** (usar booleanos) não seria adequada, pois booleanos apenas representam dois estados (verdadeiro ou falso), o que não é suficiente para representar múltiplos status como "PENDENTE", "EM_ANDAMENTO" e "CONCLUIDO".

Atividade: Integrando Enums na API de Livros com Java e Spring Boot

Objetivo:

Utilizar **enums** para gerenciar o status dos livros em uma **API de livros** criada com **Spring Boot**.

Instruções:

Crie um Enum chamado StatusLivro com os seguintes valores:

- DISPONÍVEL
- EM_EMPRESTIMO
- RESERVADO

Atualize a classe Livro para incluir um campo status do tipo StatusLivro.

Atualize o repositório e o controlador

- No repositório, permita buscar livros com base no status (por exemplo, buscar todos os livros DISPONÍVEIS).
- No controlador, crie endpoints para:
  - Criar um livro com o status definido.
  - Alterar o status de um livro (de EM_EMPRESTIMO para DISPONÍVEL, etc.).
  - Buscar livros com um determinado status.

Testando a API:

- **Criar um livro** com status `DISPONÍVEL`:

```
POST /api/livros

{

    "título": "O Senhor dos Anéis",

    "autor": "J.R.R. Tolkien",

    "status": "DISPONIVEL"

}
```

Buscar livros com status **DISPONÍVEL/PUT**:

```
GET /api/livros/status/DISPONIVEL
```

Alterar o status de um livro para `EM_EMPRESTIMO`:

```
PUT /api/livros/{id}/status
{
   "status": "EM_EMPRESTIMO"
}
```