Tópicos

1. **Revisão do Código Atual**
    - Código atual da classe `Pessoa` e seu controller.
    - A classe representa uma pessoa genérica, mas que temos diferentes tipos de pessoas no mundo real (Pessoa Física e Pessoa Jurídica).
2. **Introdução à Herança**
    - **Conceito**: Herança é um mecanismo em que uma classe herda propriedades e métodos de outra classe.
    - **Exemplo**: Pessoa Física e Pessoa Jurídica compartilham características em comum com a classe `Pessoa`.
3. **Refatoração do Projeto com Herança**
    - Criar uma classe abstrata `Pessoa` e as subclasses `PessoaFisica` e `PessoaJuridica`.
    - Justificar por que faz sentido que `Pessoa` seja uma classe abstrata e não deve ser instanciada diretamente.
4. **Criação de Classes Concretas e Controller**
    - Refatorar o controller para lidar com `/pessoa-fisica` e `/pessoa-juridica`, mostrando a flexibilidade do código.

Classe Abstrata

Uma classe abstrata é um tipo de classe que serve como modelo ou "base" para outras classes, mas que **não pode ser instanciada diretamente**. Em outras palavras, **você não pode criar objetos de uma classe abstrata**. Ela existe para ser herdada por outras classes, que devem implementar ou "completar" alguns de seus métodos.

Um dos principais motivos para usar classes abstratas é quando você quer garantir que todas as subclasses tenham certos métodos, mas deixa que cada uma implemente esses métodos de maneira específica.

Por exemplo, imagine uma classe abstrata chamada Animal, que tenha um método abstrato chamado **fazerSom**(). Cada animal (como Cachorro e Gato) vai fazer um som diferente, então cada subclasse vai implementar esse método de sua própria forma.

Resumindo, uma classe abstrata é como um "**esqueleto**" de classe: ela define o que deve existir, mas deixa para as subclasses decidirem os detalhes.

Métodos Abstratos

Um **método abstrato** é um método que é declarado, mas não tem uma implementação na classe em que foi definido. Ele serve como um "**contrato**" que todas as subclasses de uma classe abstrata **devem seguir**. Em outras palavras, o método abstrato diz: "**Essa funcionalidade precisa existir nas subclasses, mas elas devem definir o que exatamente vai acontecer**."

Por exemplo, se temos uma classe abstrata Animal com o método abstrato fazerSom(), essa classe está dizendo que todos os animais precisam fazer um som, mas o jeito como cada animal faz esse som depende da subclasse. O Cachorro pode implementar o método fazerSom() para latir, enquanto o Gato pode implementar para miar.

Então, um método abstrato **força as subclasses a implementarem** o comportamento específico, garantindo que todas elas sigam o mesmo "contrato", mas de uma maneira própria.