

Proposta de Evolução Arquitetural e Refatoração

1. Evolução da Estrutura de Armazenamento e Encapsulamento

A primeira recomendação técnica consiste na substituição do uso de arrays estáticos na classe `CadastroPessoas` por estruturas de coleções dinâmicas da API do Java, como o `ArrayList`. O uso de arrays fixos fere a escalabilidade do sistema e introduz uma complexidade desnecessária de controle de índices e verificação de transbordamento (*overflow*). Complementarmente, é imperativo aplicar o encapsulamento estrito: transformar atributos públicos ou protegidos (como observado na classe `Data`) em privados, expondo-os apenas via métodos de acesso (*getters/setters*). Isso garante a integridade dos dados e protege o estado interno dos objetos contra manipulações externas indevidas.

2. Desacoplamento de Regras de Negócio e Polimorfismo

A implementação atual das regras de tributação (3% para funcionários e 5% para gerentes) está codificada diretamente nas subclasses. Uma abordagem acadêmica superior seria a aplicação do padrão de projeto `Strategy`. Ao isolar o cálculo do imposto em uma interface ou classe de estratégia, o sistema torna-se mais flexível a mudanças na legislação tributária sem a necessidade de alterar a hierarquia de classes de `Pessoa`. Além disso, a classe `Data` poderia ser substituída pelas classes nativas do pacote `java.time` (como `LocalDate`), que oferecem tratamento robusto para anos bissextos e fusos horários, seguindo as melhores práticas da plataforma Java moderna.