Italo Ventura - Refatorações aplicadas e suas justificativas:

Princípio da Responsabilidade Única (SRP):

Refatoração: As classes foram projetadas para ter uma única responsabilidade. Exemplo, a classe GerenciadorContatos é responsável por gerenciar a lista de contatos e fornece métodos para adicionar, remover e listar contatos. Enquanto isso, a classe BuscaContatos é responsável exclusivamente pela funcionalidade de busca de contatos por nome.

Justificativa: O código fica mais coeso e fácil de entender. Cada classe tem um propósito claro e encapsula um conjunto específico de funcionalidades. Isso facilita a manutenção e evita que uma mudança em uma parte do sistema afete outras partes.

Princípio Aberto/Fechado (OCP):

Refatoração: O sistema foi projetado para ser facilmente estendido sem modificar o código existente. Por exemplo, a adição de novas funcionalidades ou a alteração dos algoritmos de busca podem ser feitas sem afetar as classes existentes.

Justificativa: Com OCP o sistema fica mais flexível e adaptável a mudanças futuras. Novas funcionalidades podem ser adicionadas sem perturbar o funcionamento, promovendo uma arquitetura robusta e escalável.

Princípio da Substituição de Liskov (LSP):

Refatoração: As classes são projetadas para serem substituíveis por suas subclasses sem afetar a funcionalidade do programa. Por exemplo, novas implementações de busca podem ser criadas sem modificar o comportamento das classes existentes.

Justificativa: Garante que as subclasses possam ser usadas de forma transparente no lugar de suas superclasses. Isso promove a reutilização do código e facilita a manutenção.

Outros que poderiam ser utulizados também seria, Encapsulamento, Dependência, Acoplamento etc.