1) Esse código fere o princípio da responsabilidade única, pois em uma classe há funções que não dizem respeito à classe em questão. tem a tendência de crescer pra sempre, ou seja, o tempo inteiro o desenvolvedor tem que pôr a mão para escrever mais código. Há um notório acoplamento, onde a classe está acoplada com várias funções de outras classes. A ideia para melhorar o código é refatorar dividindo as funções para suas devidas classes. Fere também o princípio DIP, no qual tem-se que ter uma melhor abstração.

2) **Vantagens**:
Flexibilidade: a delegação permite que você selecione quais comportamentos precisam ser reutilizados em cada instância, enquanto a herança requer que toda a classe seja estendida.
Menos acoplamento: a delegação permite que você evite herdar um grande número de métodos que você não precisa, reduzindo assim o acoplamento entre classes.
Mais fácil de manter: a delegação torna a manutenção do código mais fácil, pois as alterações em uma classe não afetam outras classes.
Desvantagens:
Mais complexidade: a delegação adiciona complexidade ao código, pois requer a criação de novas classes e métodos para delegar comportamentos.
Mais código: a delegação requer mais código do que a herança, pois você precisa criar novas classes e métodos para delegar comportamentos.
Desempenho: a delegação pode ter um impacto no desempenho, pois a chamada de método adicional pode adicionar sobrecarga. No entanto, isso depende do uso específico da delegação e das otimizações de implementação.

3) O software é algo que demanda manutenção constante, pois é um “organismo vivo”, só “morre” quando é descontinuado. E dificilmente sistemas são descontinuados. Um software com Forte Acoplamento torna-se um problema na hora de mantê-lo, devido ao alto grau de interdependência entre seus componentes.

4) Ao seguir o OCP, as classes se tornam mais flexíveis porque novos recursos podem ser adicionados à aplicação sem afetar as classes existentes. Isso permite que a aplicação seja mais facilmente adaptada a mudanças nos requisitos, sem ter que reescrever grandes partes do código.

5) O Dependency Inversion Principle (DIP) ou Princípio da Inversão de Dependência é a base para termos um projeto com um excelente design orientado a objetos, focado no domínio e com uma arquitetura flexível. A vantagem de sempre depender de classes estáveis, ou seja, abstrações, é

que elas mudam menos e facilitam a mudança de comportamento e as evoluções do código.

6) Esses dois princípios estão intimamente relacionados porque o OCP é uma maneira de garantir que o LSP seja cumprido. Quando uma classe é projetada para ser extensível, ela pode ser estendida através da criação de novas subclasses. Essas subclasses devem ser projetadas de acordo com o LSP para que possam ser usadas sem problemas no lugar da classe pai. Isso significa que, se uma classe é estendida para adicionar novas funcionalidades, as novas subclasses devem ser projetadas de tal forma que não alterem o comportamento esperado da classe pai.

   Portanto, o OCP e o LSP são dois princípios importantes da POO que trabalham juntos para garantir que o software seja flexível e extensível, sem quebrar o funcionamento correto do programa.

7) De acordo com a lei de Demeter , as classes devem conhecer e interagir com algumas outras classes possíveis. É usado para afrouxar o acoplamento, limitando a interação da classe com outras classes para fornecer estabilidade, pois um acoplamento mais rígido torna o programa difícil de manter.
8) A composição também é uma forma de extensão, só que pela delegação de trabalho para outro objeto. Diferente da herança clássica onde tipo, atributos e métodos são estendidos. Quando uma classe precisa usar o comportamento de outra, usualmente é melhor usar composição. Desta forma, uma boa maneira alternativa é usar composição.

9) O princípio Criador estabelece que uma classe deve ser responsável por criar objetos de outra classe, quando houver uma relação de agregação entre as duas classes. Isso significa que a classe que contém a outra em sua estrutura deve ser a responsável por criar objetos dessa classe contida.

   O princípio Especialista na Informação estabelece que uma determinada responsabilidade deve ser atribuída à classe que possui as informações necessárias para cumpri-la.