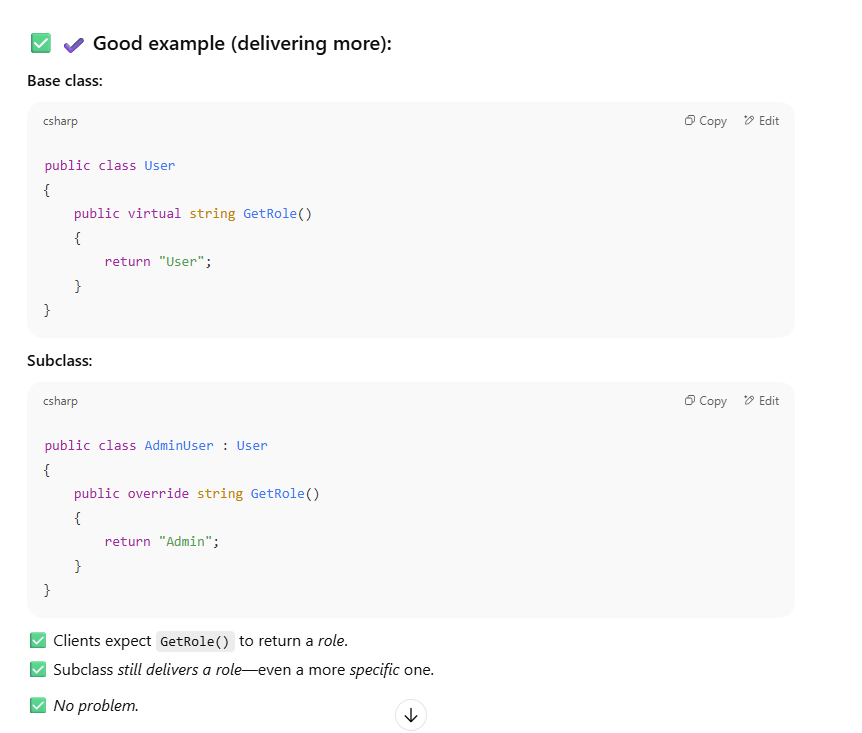
1. A derived class must be substitutable for its base class without breaking the behavior of base class.
2. subclasses **should honor the contract or promises** of their base class.  
   A screenshot of a computer

   AI-generated content may be incorrect.
3. LSP Rule says:  
   **a.** **Client code** shouldn’t need to *know or care* which subclass it has.  
   **b**. **Subclasses must not weaken preconditions** that is they should not *demand more* than the base class.  
     
   A screenshot of a chat

   AI-generated content may be incorrect.  
     
   A subclass should *never* add *extra requirements* the base didn't have. Because the Client code depending on the base *won’t know* about the extra rule. If its required then add the extra logics in the base or abstract class only.  
     
   As in the above example, PremiumPaymentProcessor add extra validations which is not in the base class that is PaymentProcessor.  
     
   ***Substitutability*** - substitute (swap in) a subclass anywhere the base class is expected, and it will still work correctly, without any pre or post conditions added in subclasses which are not in the base class.  
     
     
   **c. Subclasses must not Strengthen postconditions** that is they should not *promise less* than the base class means deliver *more* than promised is fine but cannot deliver less  
     
     
     
   A screenshot of a computer

   AI-generated content may be incorrect.
4. Its importance:  
   **a. LSP makes OCP *possible*** as without LSP, new subclasses could break existing client code which indirectly breaks OCP and then finally SRP means all thress principles are interrelated. **b. Interfaces and base classes make promises** and LSP forces to honor those promises, so client code don’t care which implementation or subclasses they use.  
   **c. Without LSP, OCP is just theory.**
5. Subtypes must **honor** the contract of their base type that’s the **heart of LSP**.
6. LSP is *only* a **design-time discipline** we have to maintain, there’s no complier error for violating LSP.
7. LSP heavily depends on **base class contract** being clear and stable, if there is any change in the base class later then all subclasses might break LSP.
8. In LSP expectation are about semantics (behavioral), not syntax therefore the compiler only enforces method existence, it doesn’t check behavioral contracts.
9. LSP relies on hum disciples, we have to design base classes *very carefully* so subclasses won’t break expectations.
10. In LSP violations can be subtle and easy to miss.