Tenha bom senso

Faça sua parte bem feita

Implemente a solução e veja se ela funciona com testes unitários

Verifique se ela pode ser otimizada

Identifique e utilize o padrão de projeto que se ajusta ao se caso

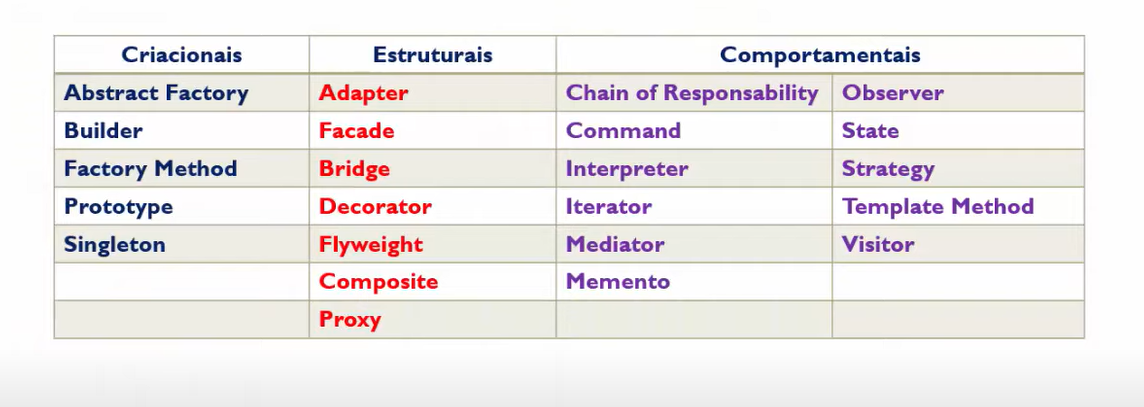
Programar para uma interface e não para uma implementação

Favorecer a composição sobre a herança

Usar os princípios SOLID

Evitar o forte acoplamento

Manter uma alta coesão entre as classes, responsabilidades bem definidas



**Criacionais** -> Abstraem ou adiam o processo de criação dos objetos, e fornecem mecanismos de criação de objetos que aumentam a flexibilidade e a reutilização de código existente.

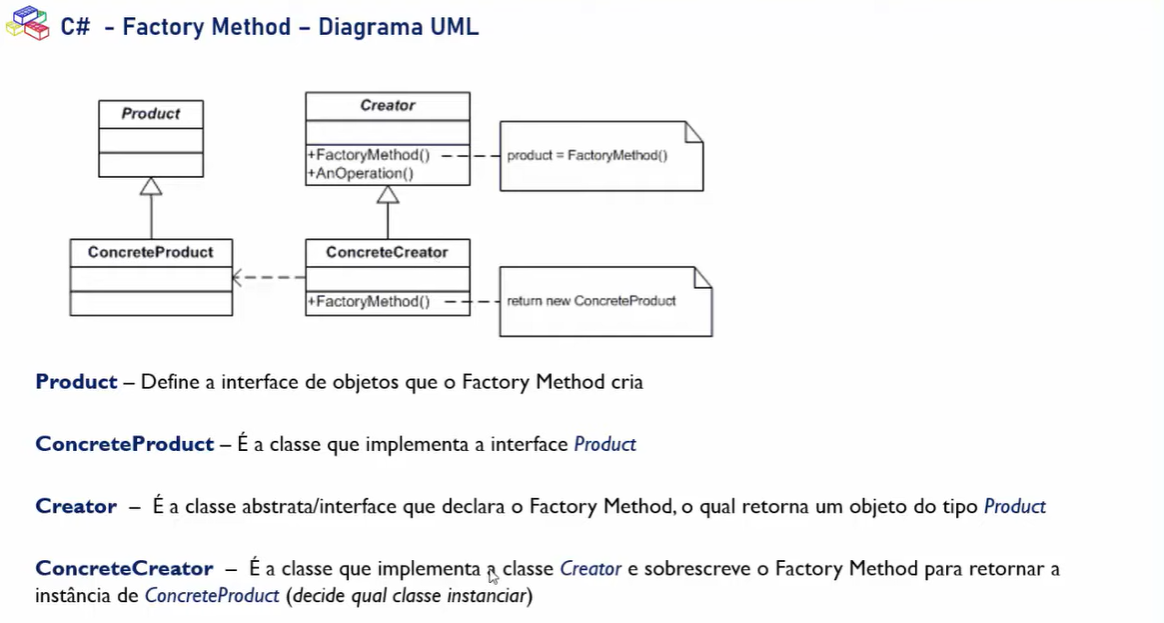
**Factory Method** -> Usado para criar objetos sem expor a lógica de criação do objeto ao cliente

Definir uma interface para criar objetos, mas permitir que as subclasses decidam qual classe instanciar

Permitir que Cliente não precise saber quais tipos de objetos serão criados

Usa herança e depende de uma subclasse para tratar com a instanciação do objeto desejado

Cria uma instancia de varias classes derivadas



Beneficios

