Padrões de Projeto: Builder e Prototype

1. Builder

Contexto
Em sistemas orientados a objetos, muitas vezes é necessário criar objetos complexos compostos por diversos atributos opcionais e obrigatórios. O processo de construção pode se tornar confuso quando há muitos parâmetros.

Problema
Se utilizarmos apenas construtores tradicionais ou métodos setters, a criação de objetos grandes pode se tornar verbosa e suscetível a erros, especialmente quando muitos parâmetros são opcionais.

Solução
O padrão Builder propõe a separação do processo de construção de um objeto da sua representação. Dessa forma, é possível criar diferentes representações de um objeto passo a passo, garantindo clareza e flexibilidade.

2. Prototype

Contexto
Em algumas situações, a criação de objetos pode ser custosa em termos de tempo ou recursos. Isso ocorre quando os objetos possuem configurações complexas ou dependem de processos caros de inicialização.

Problema
Se for necessário criar várias instâncias semelhantes, recriar cada objeto do zero pode gerar desperdício de recursos e reduzir o desempenho do sistema.

Solução
O padrão Prototype sugere a criação de novos objetos a partir da clonagem de instâncias já existentes. Assim, é possível reutilizar configurações de objetos base, economizando tempo e esforço de criação.