

Design Pattern

Mateus Barbosa Martins

Builder

O Design Pattern Builder é uma solução elegante para construir objetos complexos com uma interface simples. Ele separa a construção de um objeto de sua representação, permitindo que você crie diferentes variações de um objeto a partir de uma única interface.



Simplicidade



Flexibilidade



Reutilização

Builder

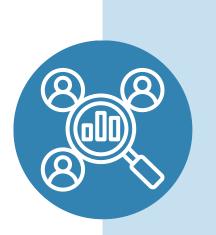
Imagine que você está construindo um sistema de ecommerce e precisa criar um objeto Produto com atributos complexos, como nome, preço, descrição, imagem, etc. O Builder pode ser usado para facilitar este processo:

- O Builder encapsula a lógica de construção do objeto, tornando o código mais organizado.
- O uso de métodos de configuração permite uma construção passo a passo, adaptando-se às necessidades.
- O Builder pode ser usado para criar múltiplos objetos com configurações diferentes, sem a necessidade de criar classes separadas para cada variação.

```
1 class Produto:
        def __init__(self, nome, preco, descricao, imagem):
            self.nome = nome
            self.preco = preco
            self.descricao = descricao
            self.imagem = imagem
 8 class ProdutoBuilder:
        def __init__(self):
           self.produto = Produto("", 0, "", "")
11
12
       def set_nome(self, nome):
            self.produto.nome = nome
           return self
15
        def set_preco(self, preco):
            self.produto.preco = preco
           return self
        def set_descricao(self, descricao):
21
            self.produto.descricao = descricao
           return self
       def set_imagem(self, imagem):
25
            self.produto.imagem = imagem
           return self
       def build(self):
            return self.produto
31 # Usando o builder para criar um produto
32 produto = ProdutoBuilder() \
        .set_nome("Camiseta") \
        .set_preco(29.99) \
        .set_descricao("Camiseta de algodão, tamanho M") \
        .set_imagem("camiseta.jpg") \
        .build()
    print(produto.nome) # Saída: Camiseta
40 print(produto.preco) # Saída: 29.99
```

Prototype

O Design Pattern Prototype é uma solução para criar novos objetos a partir de um modelo já existente (o protótipo), sem precisar reescrever todo o processo de criação. Ele é ideal para situações onde a criação de novos objetos é complexa ou requer muitas etapas.



Reutilização



Eficiência



Simplicidade

Prototype

Imagine que você está criando um sistema de gerenciamento de usuários e precisa criar novos usuários com base em um perfil padrão. O Prototype pode ser usado para clonar o perfil padrão e, em seguida, ajustar as propriedades do novo usuário:

- O Prototype encapsula a lógica de criação de novos objetos, simplificando o processo.
- O método clone() fornece uma forma eficiente de criar cópias do objeto, sem a necessidade de reescrever todo o código.
- O Prototype permite criar diferentes variações de um objeto a partir de um modelo base, ajustando apenas as propriedades necessárias.

```
class Usuario:
        def __init__(self, nome, email, senha):
            self.nome = nome
            self.email = email
            self.senha = senha
        def clone(self):
            return Usuario(self.nome, self.email, self.senha)
    perfil_padrao = Usuario("Padrão", "padrao@example.com", "123456")
12
   # Criando um novo usuário a partir do perfil padrão
   novo usuario = perfil padrao.clone()
   novo usuario.nome = "João da Silva"
   novo_usuario.email = "joao.silva@example.com"
    print(novo usuario.nome) # Saída: João da Silva
19 print(novo_usuario.email) # Saída: joao.silva@example.com
```