Builder

Propósito

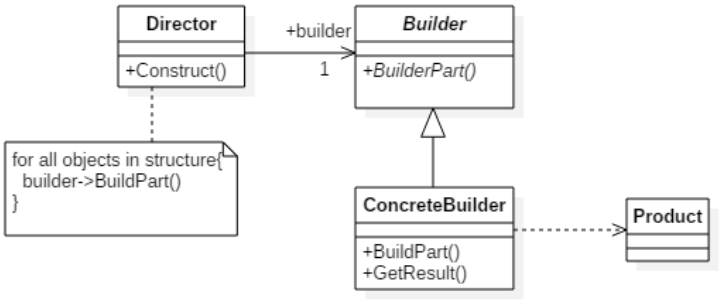
Separa la construcción de un objeto complejo de su representación, de modo que el mismo proceso de construcción se utiliza para crear diferentes representaciones.

Aplicabilidad

El Builder se utiliza cuando:

- El algoritmo para crear un objeto complejo debería ser independiente de las partes que lo componen y de la forma en que se ensamblan
- El proceso de construcción debe permitir diferentes representaciones de los objetos que se construyen

Estructura



Participantes

- Builder: especifica una interface abstracta para crear partes de un Producto
- ConcreteBuilder:
    - Implementa la interface del Builder para construir y ensamblar partes del producto
    - Define y guarda la representación que crea
    - Proporciona una interface para recuperar el producto
- Director: construye un objeto utilizando la interface del Builder
- Product:
    - Representa el objeto complejo que se construye. ConcreteBuilder construye la representación interna del producto y define el proceso que se encarga del ensamblado
    - Incluye las clases que definen las partes del objeto complejo

Colaboraciones

- El cliente crea el objeto Director y lo configura con el Builder que se desea
- El Director indica al Builder la parte del producto que toca construir
- El Builder maneja las peticiones del director y agrega las partes al producto
- El cliente obtiene el producto del Builder

Consecuencias

Algunas consecuencias clave del patrón Builder:

1. Permite variar la representación interna del producto: El objeto Builder proporciona al director una interface para construir el producto. La interface permite al builder ocultar la representación y la estructura interna del producto. También oculta el modo en que se ensambla el producto. Como el producto se construye mediante una interface, lo único que hay que hacer para cambiar la representación interna del producto es definir un nuevo tipo de builder.
2. Aísla el código de la construcción y la representación. El patrón Builder mejora la modularidad al encapsular el modo en que se construye y representa un objeto complejo. Los clientes no necesitan conocer las clases que definen la estructura interna del producto, esas clases no aparecen en la interface del Builder.
   Cada ConcreteBuilder contiene todo el código para crear y ensamblar un tipo particular de producto. El código se escribe una vez, y después se puede reutilizar el Director para crear variantes del producto.
3. Proporciona un control más fino del proceso de construcción. A diferencia de los otros patrones creacionales que construyen productos de una vez, el patrón Builder construye el producto paso a paso bajo el control del director. Solo cuando se ha terminado de fabricar el producto, el director lo recupera del Builder. Por eso la interface del Builder refleja el proceso de construcción del producto mucho más que los otros patrones creacionales. Esto proporciona un control más fino del proceso de construcción y, consecuentemente, de la estructura interna del producto resultante.