Builder

Nombre:
Builder

Clasificación del patrón:
Creacional

Intención:
Separa la construcción de un objeto complejo de su representación, hace que el mismo proceso de construcción pueda crear varias representaciones.

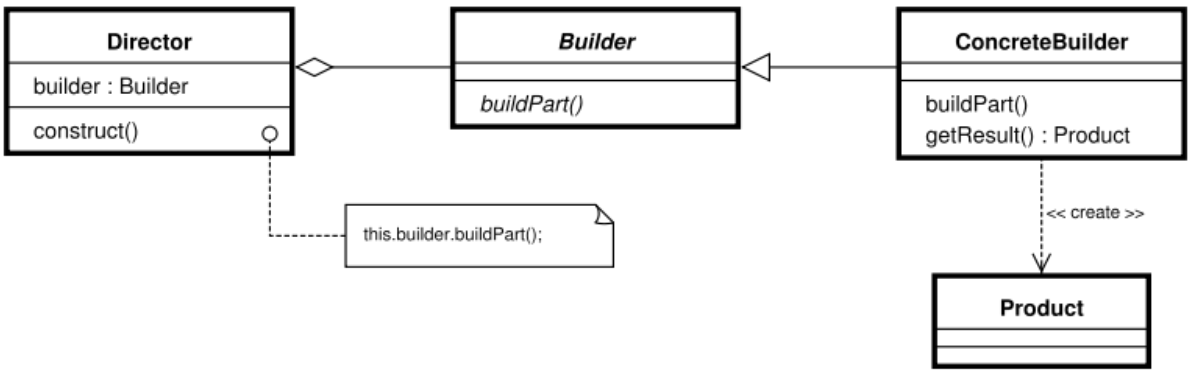
Otros nombres:
Constructor

Motivación:
En el caso de un editor de texto, frecuentemente el lector tendrá que interpretar y/o convertir un tipo de documento a otro lenguaje. El problema es que no se sabe con exactitud cuántos de esos lenguajes debe interpretar el editor. Es necesaria entonces una estructura que permita añadir una especificación sin modificar el lector como tal. Aquí nace el patrón Builder.

Aplicabilidad:
Se debe usar cuando:

- Se requiere tener un control detallado sobre el proceso de construcción de un objeto.
- Es necesario construir un objeto por pasos.

Estructura:



Participantes:

- Constructor: Especifica una interfaz abstracta para construir partes del objeto producto.
- Constructor Concreto: Implementa la interfaz constructor ensamblando y construyendo las partes del producto.
- Director: Construye un objeto a través de constructor.
- Producto: Representa el objeto complejo a construir.

Colaboraciones:

- El cliente crea el objeto director y especifica las partes que necesita del Constructor.
- El director notifica al Constructor las partes requeridas.
- Constructor maneja la petición y añade las partes al producto.
- El cliente obtiene el producto de Constructor.

Ventajas:

- El director crea el producto a través de una interfaz abstracta, lo que hace posible la creación de un nuevo tipo de objeto dentro del mismo proceso de construcción.
- Diferentes representaciones de el producto pueden ser construidas añadiendo un nuevo director.
- La construcción del proceso puede ser controlada porque el objeto es construido en pasos por el director. Se tiene un control detallado del proceso de construcción.

Desventajas:

- Se requiere más dominio del conocimiento por parte del cliente en la construcción de objetos.

Implementación:

- Construcción de la interfaz y ensamblaje: Los constructores crean los objetos bajo un modelo paso a paso. La interfaz debe estar bien generalizada para permitir la construcción de todo tipo de objetos.
- ¿Por qué no usar clases abstractas en el producto?: A menudo los productos generados a partir de constructores concretos se diferencian mucho de su representación y el manejo se hace difícil con clases abstractas.
- Definir métodos vacíos por defecto en el Constructor.

Código de ejemplo:

```
package Builder01;
public class Main
{
    public static void main(String[] args)
    {
        // Crear el objeto Director
         Director objFabrica = new Director();
        // Crear los objetos ConcreteBuilder
         BuilderCoche base  = new ConstructorCocheBase();
         BuilderCoche medio = new ConstructorCocheMedio();
         BuilderCoche full  = new ConstructorCocheFull();
        // Construir un coche con equipamiento base
         objFabrica.construir( base );
         Coche cocheBase = base.getCoche();
        // Construir un coche con equipamiento medio
        objFabrica.construir( medio );
        Coche cocheMedio = medio.getCoche();
        // Construir un coche con equipamiento full
        objFabrica.construir( full );
        Coche cocheFull = full.getCoche();
        // Mostrar la información de cada coche creado
        mostrarCaracteristicas( cocheBase );
        mostrarCaracteristicas( cocheMedio );
        mostrarCaracteristicas( cocheFull );
    }
    // ------------------------------
    public static void mostrarCaracteristicas( Coche coche )
    {
        System.out.println( "Motor: " + coche.getMotor() );
        System.out.println( "Carrocería: " + coche.getCarroceria() );
        System.out.println( "Elevalunas eléctrico: " + coche.getElevalunasElec() );
        System.out.println( "Airea acondicionado: " + coche.getAireAcond() );
        System.out.println("====================================");
    }
}
```

Usos conocidos:

- Fue patrón común de SmallTalk
- Aplicaciones de procesamiento de texto RTF

Patrones relacionados:

- Abstract Factory
- Composite

Bibliografía:

No específico. (No específico). GoF Design Patterns (Versión 2.1.0) [Aplicación móvil]. Descargado de: https://drive.google.com/file/d/0BywiVyFlIabXcVhGZlJBcnhWTkU/view.

Patrones de Diseño Software [Página web]. (s.f.). Ubicación https://informaticapc.com/patrones-de-diseno/builder.php.

Junta de Andalucía. (s.f). Constructor. Marco de Desarrollo de la Junta de Andalucía. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/185.