Patrones de Diseño: Patrones de Creación. Tema 3-4: Factory Method

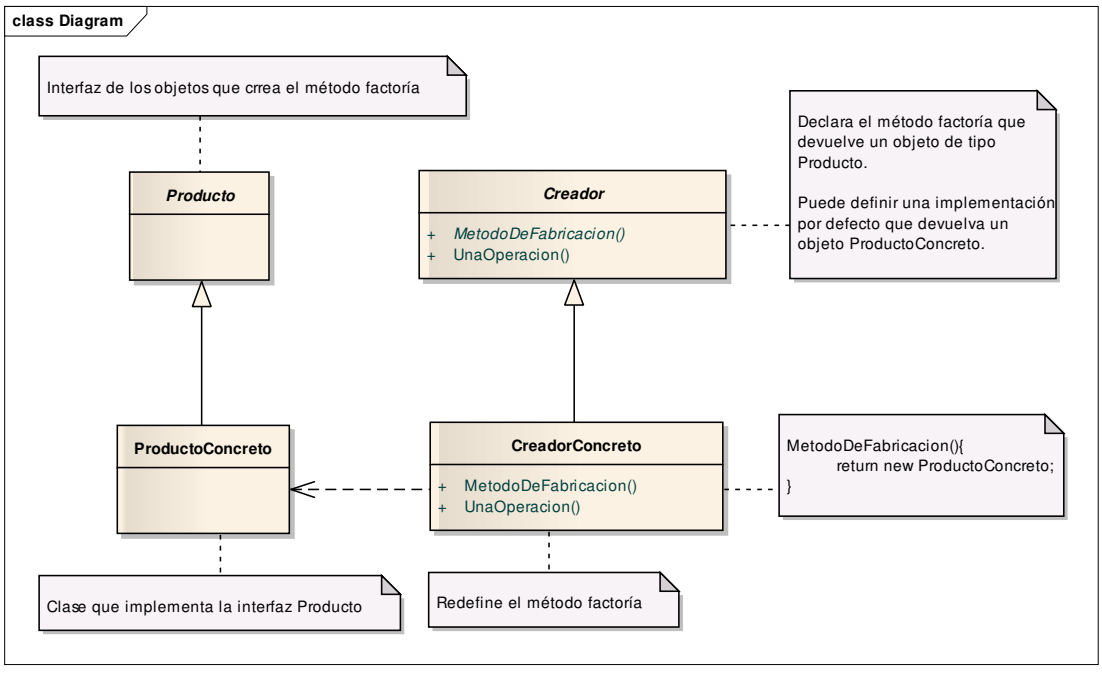
Descripción del patrón

- **Nombre:**
  - Método de Fabricación
  - También conocido como Virtual Builder o Virtual Constructor (Constructor virtual)
- **Propiedades:**
  - Tipo: creación
  - Nivel: clase
- **Objetivo o Propósito:**
  - Define una interfaz para crear un objeto pero deja que sean las subclases quienes decidan que clase instanciar. Permite que una clase delegue en sus subclases la creación del objeto.

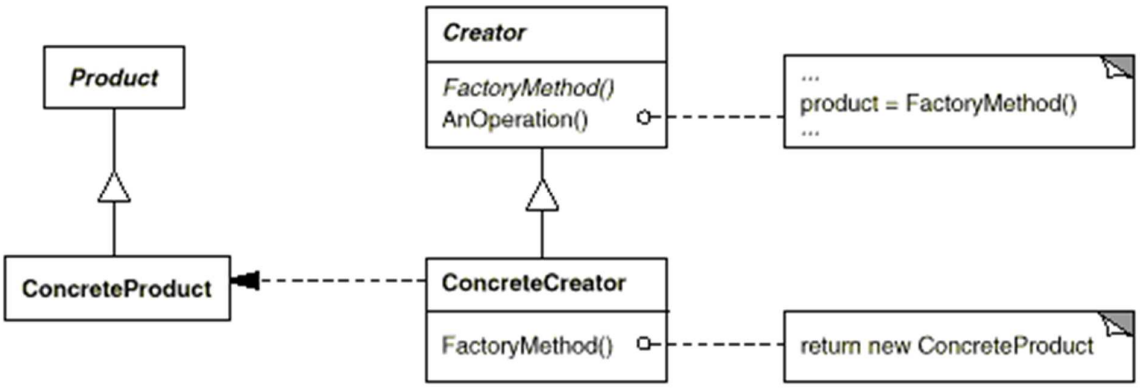
Aplicabilidad

- Use el patrón Factory Method cuando:
  - Una clase no puede prever la clase de objetos que debe crear.
  - Una clase quiere que sean sus subclases quienes especifiquen los objetos que ésta crea.
  - Se sabe cuándo crear un objeto pero no conoce su tipo.
  - Se desea crear un framework extensible.
  - Necesita algunos constructores sobrecargados con la misma lista de parámetros (no permitido en Java). En vez de eso, utilizar varios métodos de fabricación con distinto nombre.

Estructura



Estructura



Estructura. Participantes

- **Producto:** Define la interfaz de los objetos que crea el método de fabricación.
- **ProductoConcreto**: Implementa la interfaz Producto.
- **Creador:** Declara el método de fabricación, el cual devuelve un objeto de tipo Producto. También puede definir una implementación predeterminada del método de fabricación que devuelva un objeto ProductoConcreto.
- **CreadorConcreto:** Redefine el método de fabricación para devolver una instancia de un ProductoConcreto.

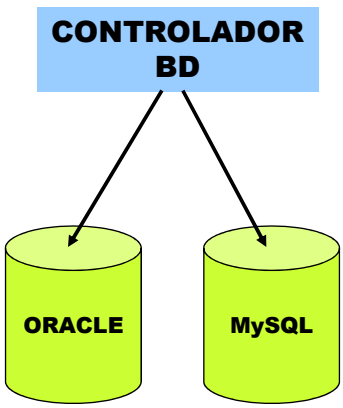
Estructura. Variaciones

- **Variaciones del patrón:**

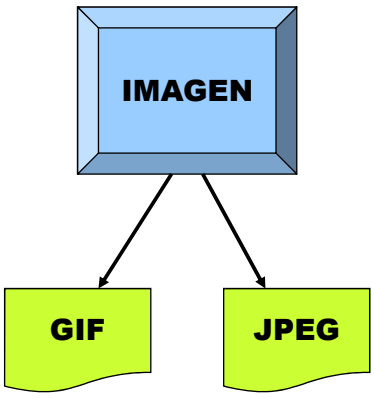
- **Creador:** Puede proporcionar una implementación estándar para los métodos de fabricación. Por ello, Creador no tiene que ser una clase abstracta o una interfaz, sino una clase completamente definida.
- **Producto:** Puede ser implementado como una clase abstracta. Debido a que Producto ya es una clase, puede incluir implementaciones para los otros métodos.
- El método de fabricación puede tomar algún parámetro para poder crear varios tipos de productos. Esto reduce el número de métodos de fabricación necesarios.

Ejemplos

- **JDBC**



- **Imágenes**



Consecuencias

1. **Proporciona enlaces para las subclases.** Crear objetos dentro de una clase con un método de fabricación es siempre más flexible que hacerlo directamente. El Factory Method da a las subclases un punto de enlace para proveer una versión extendida de un objeto.

2. **Conectar jerarquías de clases paralelas.** Las jerarquías de clases paralelas se producen cuando una clase delega alguna de sus responsabilidades a una clase separada. Los métodos factoría pueden ser llamados por los clientes, no sólo por los Creadores.
   - Ejemplo: Manipuladores de figuras: un tipo de manipulador con varios métodos factoría para cada tipo de figura

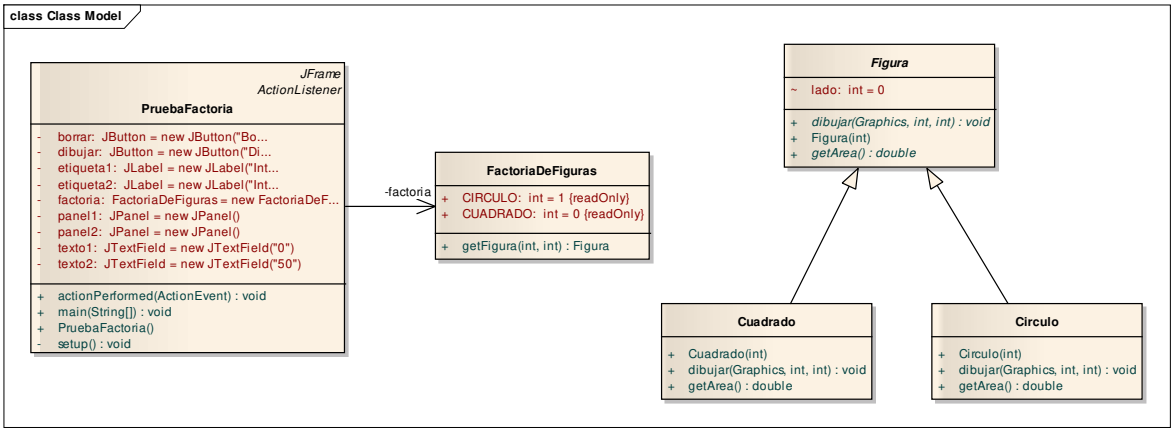
Patrones relacionados

- **Prototipo**. Factory Method es un patrón de creación de clase, esto quiere decir que utiliza la herencia para decidir qué objeto instanciar. Mientras que el Prototipo es un patrón de creación de objeto porque delega la instanciación a otro objeto. En contraste, Prototipo usa, solo una clase "factory" de creación (encapsulada en el prototipo) y permite la creación dinámica de objetos.
- **Abstract Factory**. El patrón Factory Method proporciona una interfaz para producir objetos, pero el tipo exacto de objeto que produce depende de qué subclase exacta es usada por el cliente. Es similar a Abstract Factory en que los métodos de Abstract Factory pueden ser implementados como métodos de factoría. La principal diferencia es que mientras Abstract Factory trata con una familia de productos, el Factory Method solo trata con un único producto**. El Factory Method ofrece una interfaz para crear un objeto, no una familia de objetos relacionados, como en Abstract Factory.**

Código de ejemplo

Dibujo de Figuras

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Producto: Figura.
  - ProductoConcreto: Cuadrado, Circulo.
  - Creador: FactoriaDeFiguras.
- Nos encontramos en este ejemplo con la variación del patrón en la que el Creador proporciona una implementación estándar para los métodos de fabricación. Creador no es una clase abstracta o una interfaz, sino una clase completamente definida (ver transparencia 7: Estructura. Variaciones).