



Factory Method


DISEÑO Y ARQUITECTURA DE SOFTWARE

CLAUDIA ALEJANDRA SALDIVAR TORRES



Factory Method

El patrón “Factory Method” se define como una interfaz para la creación de cierto tipo de objeto, permitiendo que las subclases decidan que clase concreta necesitan instanciar. El problema que se plantea en algunos entornos es que una clase no puede anticipar el tipo de objetos que debe crear debido a la jerarquía de clases existente, lo cual provoca que tenga que delegar esta tarea en una subclase.




Viene a solucionar el problema que se presenta cuando tenemos que crear la instancia de un objeto pero a priori no sabemos aún que tipo de objeto tiene que ser, generalmente, porque depende de alguna opción que seleccione el usuario en la aplicación o porque depende de una configuración que se hace en tiempo de despliegue de la aplicación.



Este patrón se compone de:

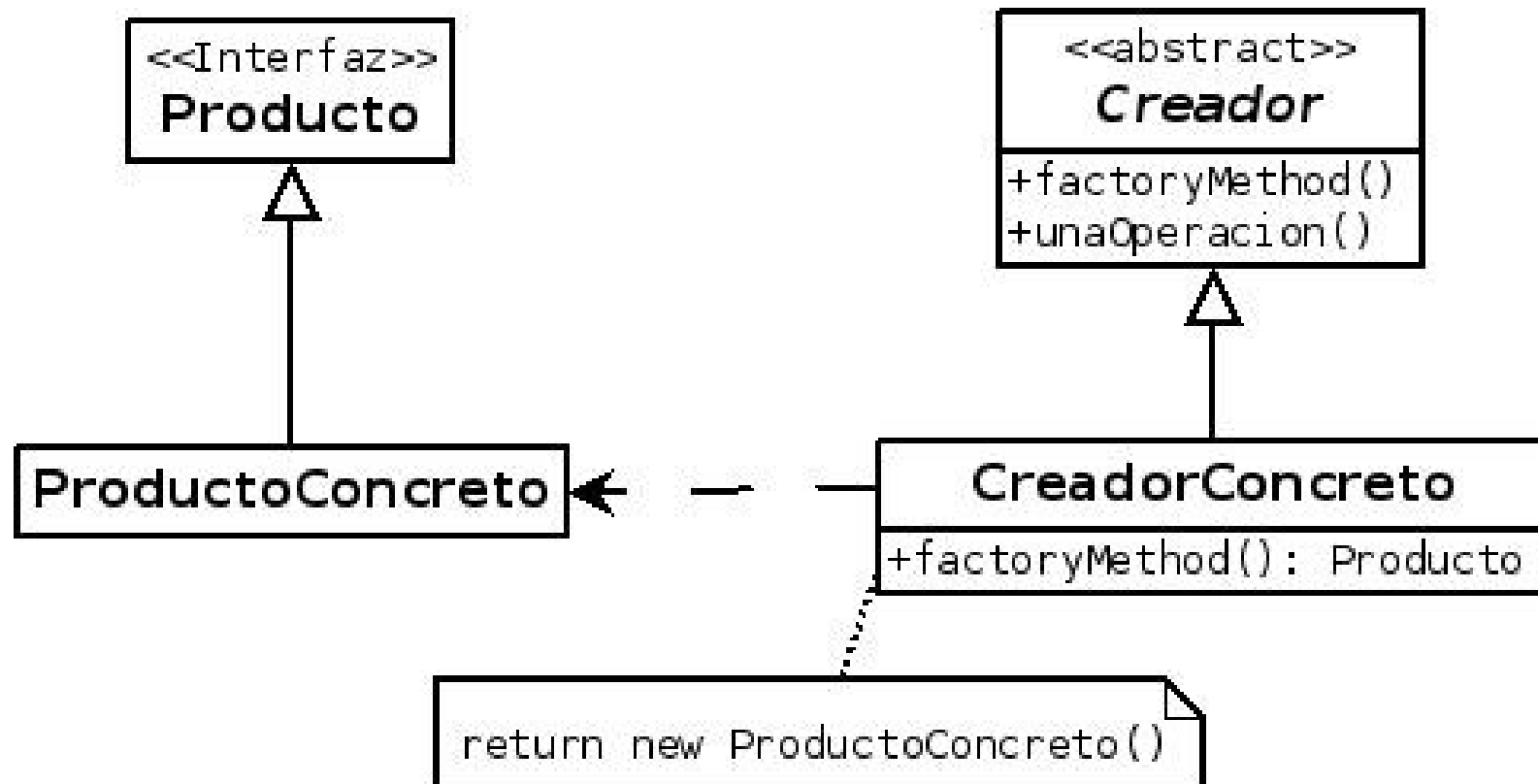
- **Product:** Define la interfaz de los objetos que se van a crear.
- **ConcreteProduct:** Implementación de la interfaz.
- **Creator:** Declara la factoría y devuelve un objeto del tipo producto. Además, puede definir una implementación por defecto para la factoría que devolvería un objeto **ConcreteProduct** por defecto.
- **ConcreteCreator:** Sobreescribe el método de la factoría para devolver una instancia concreta de un objeto **ConcreteProduct**.



A la hora de la implementación tenemos varias opciones disponibles:

- El Creator es una clase abstracta y no ofrece ningún tipo de implementación para la factoría que declara.
- El Creator es una clase concreta y ofrece una implementación por defecto de la factoría.
- La parametrización de la factoría. Donde la factoría recibe un parámetro con el tipo de objeto que tiene que crear. Todos los objetos disponibles tiene que compartir la interfaz Product.

Estructura



La solución para esto es hacer un método abstracto (el método de la fábrica) que se define en el creador. Este método abstracto se define para que devuelva un producto. Las subclases del creador pueden sobrescribir este método para devolver subclases apropiadas del producto...

Ventajas y Desventajas

- **Como ventajas se puede citar:**
- Elimina la necesidad de instanciar de forma explícita los objetos que se van a utilizar, lo que tiene ventajas claras en el desacoplamiento e interdependencia de las clases.
- Es fácilmente extendible ya que la arquitectura queda abierta a desarrollos horizontales con nuevas clases que extiendan a la factoría y a la familia de productos.
- Permite encapsular en las clases factorías toda la lógica de creación de objetos, que a pueden ser más complejas que realizar un simple new.
- Es una buena forma de disponer de una solución en la arquitectura de la aplicación que se entiende fácilmente con metáforas.
- **Como desventajas podemos considerar las siguientes:**
- Aumenta la complejidad de la aplicación, añadiendo nuevos niveles de indirección. No obstante, al tratarse de un patrón conocido es fácil abstraerse de esta complejidad
- No se puede utilizar cuando el cliente no tiene claro qué tipo concreto de objeto de la familia necesita.
- Al delegar funciones puede ser más complejo encontrar en primera instancia la mecánica de funcionamiento de la aplicación.
- Uso no es adecuado en todos los casos: es necesaria una familia de objetos sobre la que operar. Si no existe este polimorfismo no se tiene por qué caer en la tentación de hacer factorías de objetos para al futuro. Hay que analizar bien el problema al que nos enfrentamos para ver si encaja.