# Abstract Factory

## Propósito

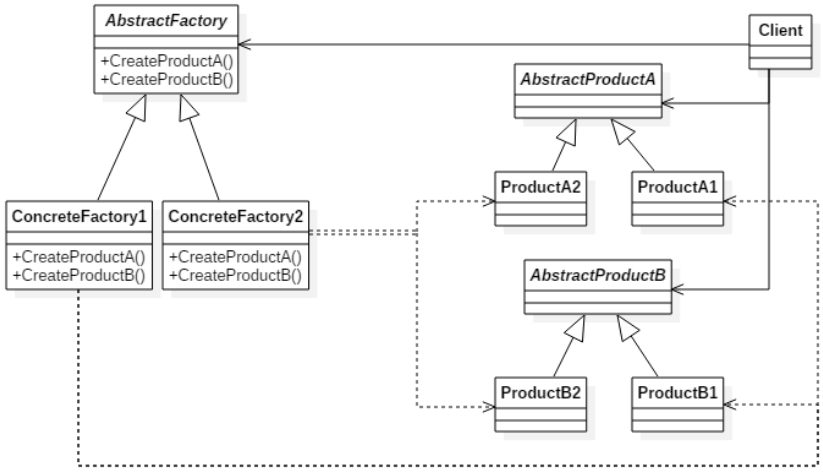
Proporciona una interface para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas.

## Aplicabilidad

El patrón Abstract Factory se utiliza cuando:

- Un sistema debe ser independiente de cómo se crean, componen y representan sus productos
- Un sistema debe ser configurado con una o varias familias de productos
- Tenemos una familia de productos relacionados que deben ser utilizados conjuntamente y queremos asegurarnos de que sea así
- Se desea proveer una librería de productos y solo se quiere revelar las interfaces y no las implementaciones

## Estructura




## Participantes

- AbstractFactory: declara una interface que tiene las operaciones para crear productos
- ConcreteFactory: implementa las operaciones para crear objetos producto
- AbstractProduct: declara una interface de un tipo de producto
- ConcreteProduct:  define un producto que será creado con el correspondiente factory
- Client: utiliza las interfaces declaradas por las clases AbstractFactory y AbstractProduct

## Colaboraciones

- Normalmente solamente se crea una instancia de ConcreteFactory en tiempo de ejecución. Esta factoría concreta crea productos de una implementación particular. Para crear otro tipo de productos hay que usar otra factoría
- El AbstractFactory difiere la creación de productos a sus subclases concretas.

## Consecuencias

El patrón Abstract Factory tiene los siguientes beneficios e inconvenientes:

1. Aísla las clases concretas. El patrón AF nos ayuda a controlar los tipos de objetos que crea una aplicación. Como la factoría encapsula la responsabilidad y el proceso de crear productos, aísla a los clientes de las implementaciones. Los clientes manipulan las instancias a través de las interfaces abstractas. Los nombres de las clases producto no aparecen en el código del cliente
2. Facilita el intercambio de familias de productos. La clase de una factoría concreta aparece solamente una vez en una aplicación, cuando se instancia. Esto hace sencillo cambiar la factoría concreta que se usa. Se puede utilizar una configuración diferente con solo cambiar la factoría concreta. Como una factoría abstracta crea una familia completa de productos, se cambia de golpe toda la familia.
3. Promueve la consistencia entre productos. Cuando se diseñan un conjunto de productos para trabajar juntos, es importante garantizar que una aplicación usa una familia a la vez. El AF nos facilita cumplir esta restricción
4. Soportar nuevos tipos de productos puede ser difícil. Extender las factorías abstractas para producir nuevos tipos de productos no es sencillo. Esto se debe a que el Abstract Factory fija el conjunto de productos que se puede crear. Soportar nuevos tipos de productos supone cambiar la clase AbstractFactory y todas sus subclases. En la sección Implementación se discute una solución a este problema.