PATRÓN FACTORY METHOD

PROPÓSITO

- ***Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar***

- ***Permite que una clase delegue en sus subclases la creación de objetos***

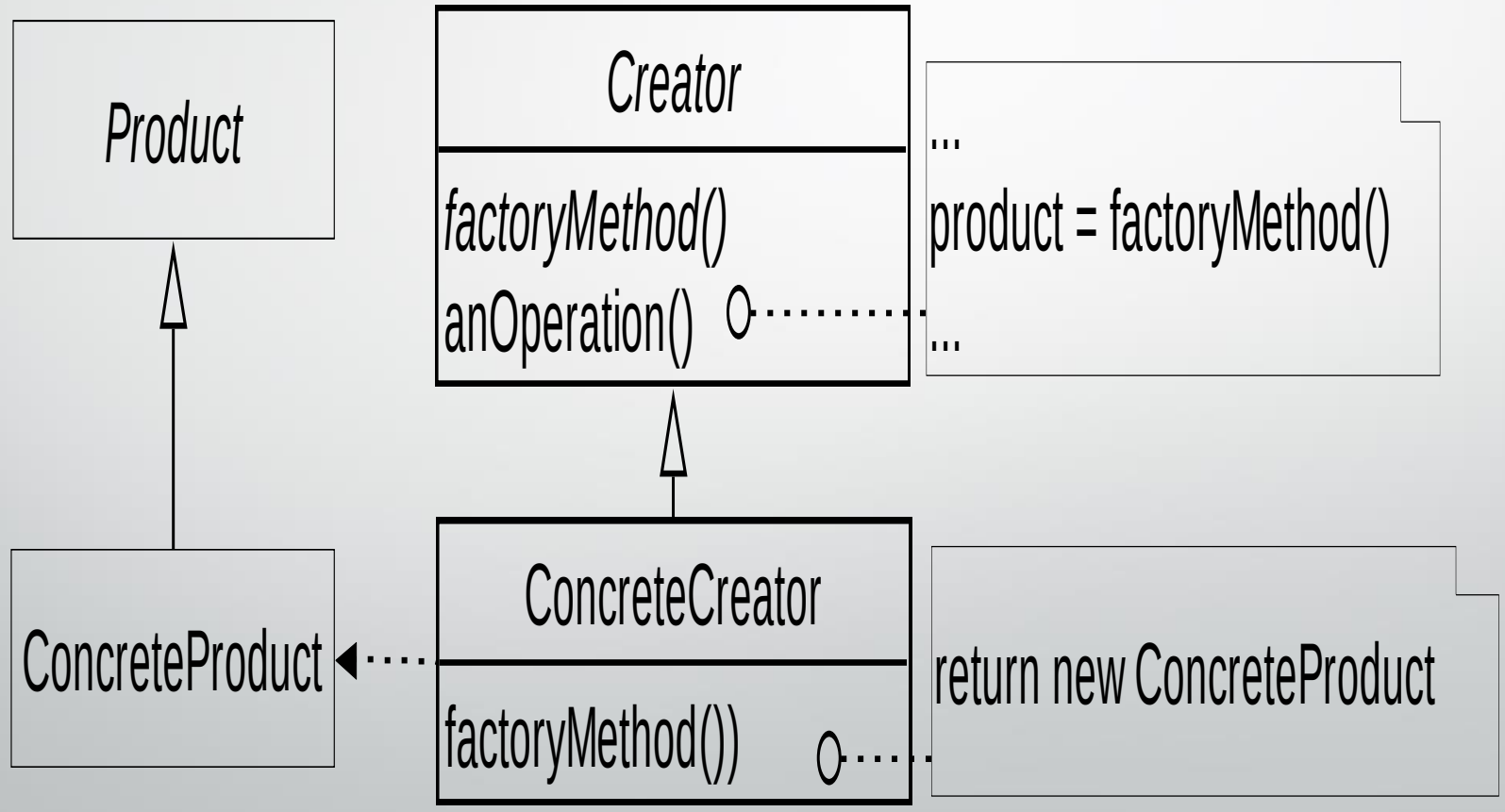
APLICABILIDAD

- 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
- Las clases delegan la responsabilidad en una de entre varias clases auxiliares, y queremos localizar en una subclase dicha delegación

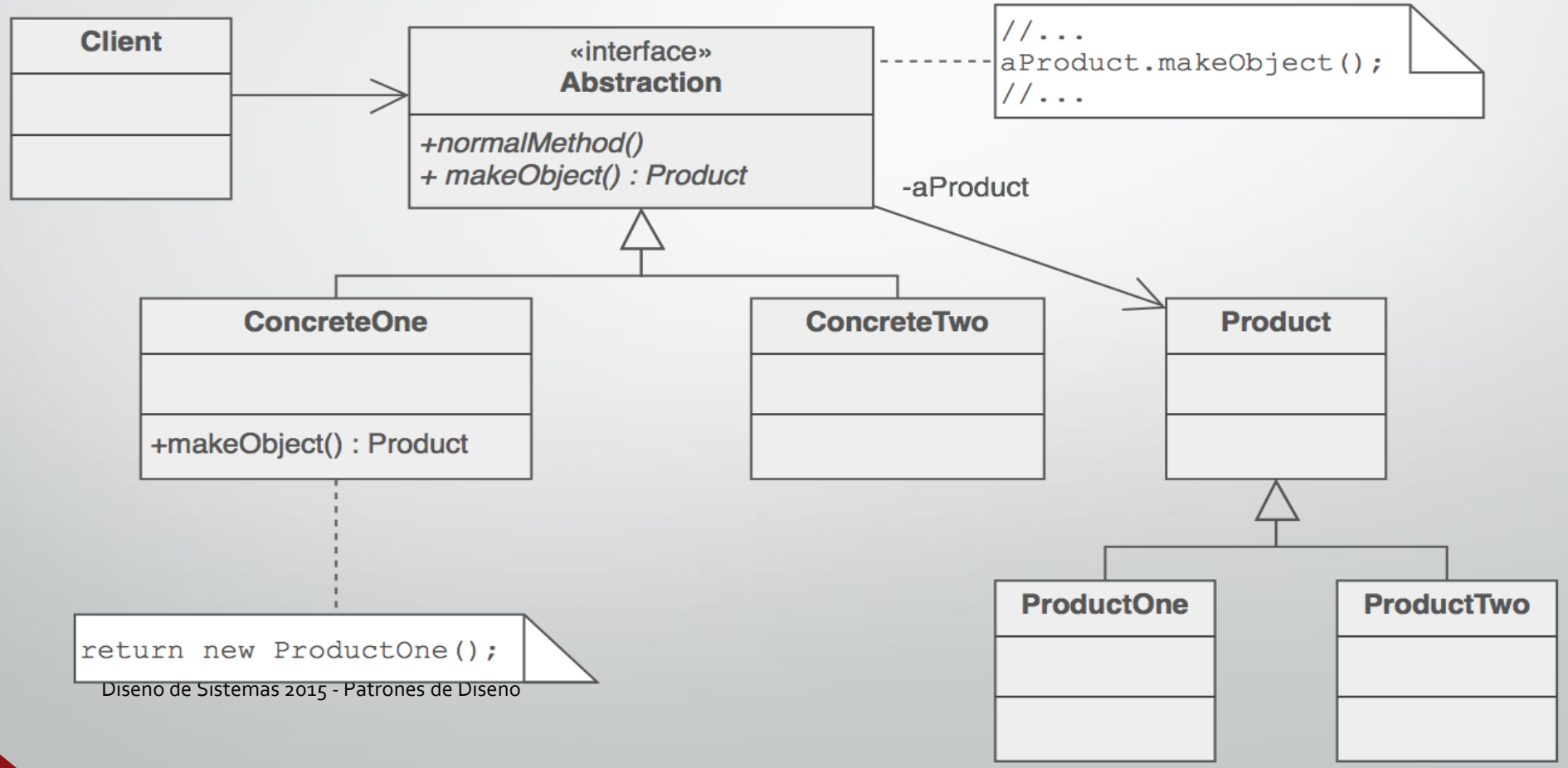
✓ Solución

Definir una superclase que tenga una interface para crear un objeto, dejando en las subclases la responsabilidad de decidir que clase instanciar

ESTRUCTURA


Product
Creator
factoryMethod()
anOperation()
...
product = factoryMethod()
...
ConcreteProduct
ConcreteCreator
factoryMethod())
return new ConcreteProduct

El cliente está totalmente desacoplado de los detalles de implementación de las clases derivadas. Creación polimórfica es ahora posible.



PARTICIPANTES

Las clases y / u objetos que participan en este patrón

(con ejemplos)

Del producto (Página)
define la interfaz de los objetos creados por el método de fábrica

ConcreteProduct (SkillsPage, EducationPage, ExperiencePage)
implementa la interfaz del producto

Creador (Documento)
declara el método de fábrica, que devuelve un objeto del tipo de producto. **Creator** también puede definir una implementación predeterminada del método de fábrica que devuelve un objeto ConcreteProduct defecto.
puede llamar al método de fábrica para crear un objeto de producto.

ConcreteCreator (informe, Montevideo)
anula el método de fábrica para devolver una instancia de un ConcreteProduct.

VENTAJAS

- **El Creador se apoya en sus subclases para definer el método de fabricación de manera que éste devuelve una instancia del ProductoConcreto apropiado.**
- **Proporciona enganches para las subclases, para proveer una versión extendida de un objeto.**
- **Conecta jerarquías de clases paralelas, se produce cuando una clase delega alguna de sus responsabilidades a una clase separada.**

DESVENTAJAS

Un inconveniente potencial de los métodos de fabricación es que los clientes pueden tener que heredar de la clase Creador simplemente para crear un determinado objeto ProductoConcreto

PATRONES RELACIONADOS

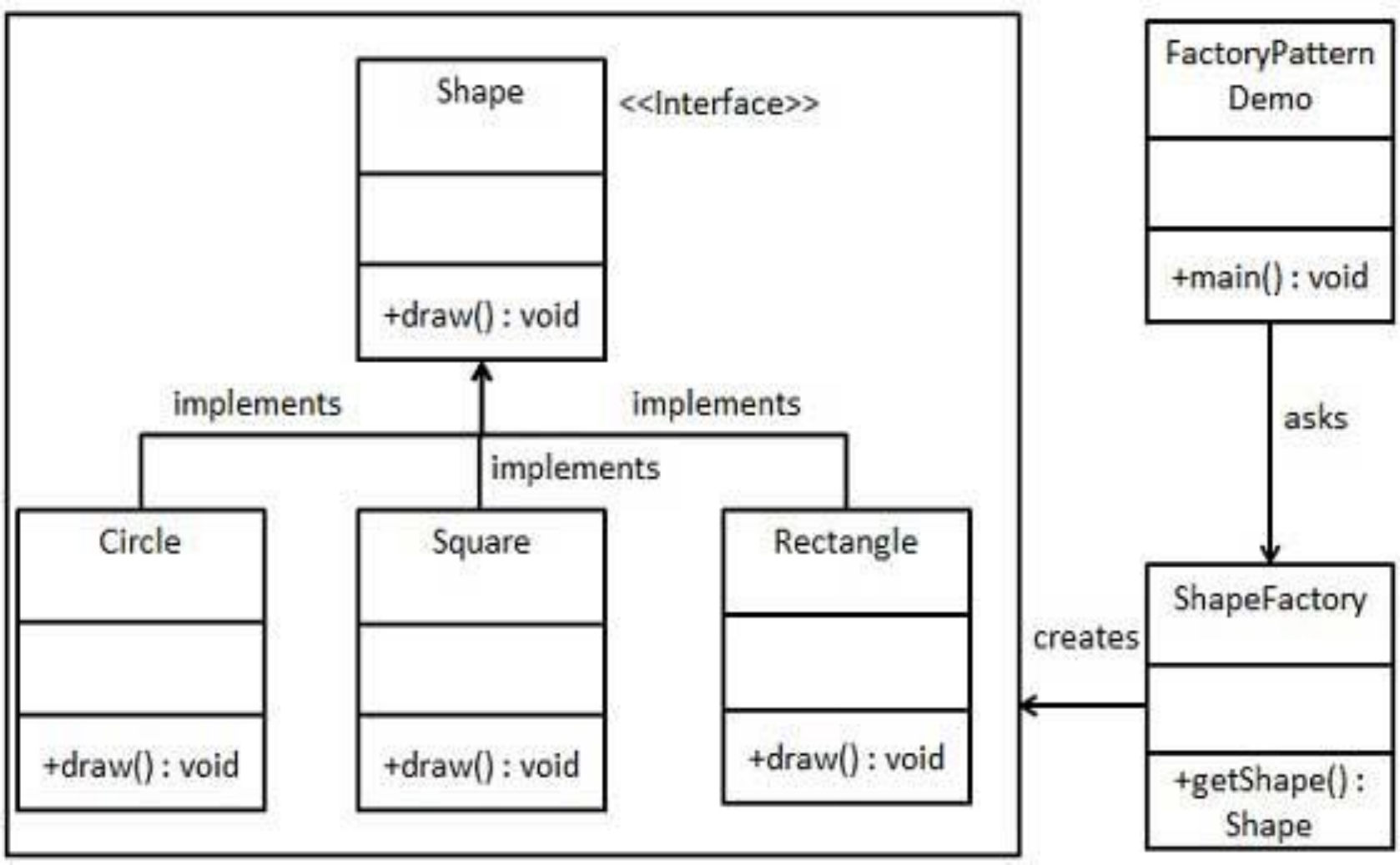
- ✓ **Con el patrón Abstract Factory**

DIFERENCIAS ENTRE ABSTRACT FACTORY Y FACTORY METHOD

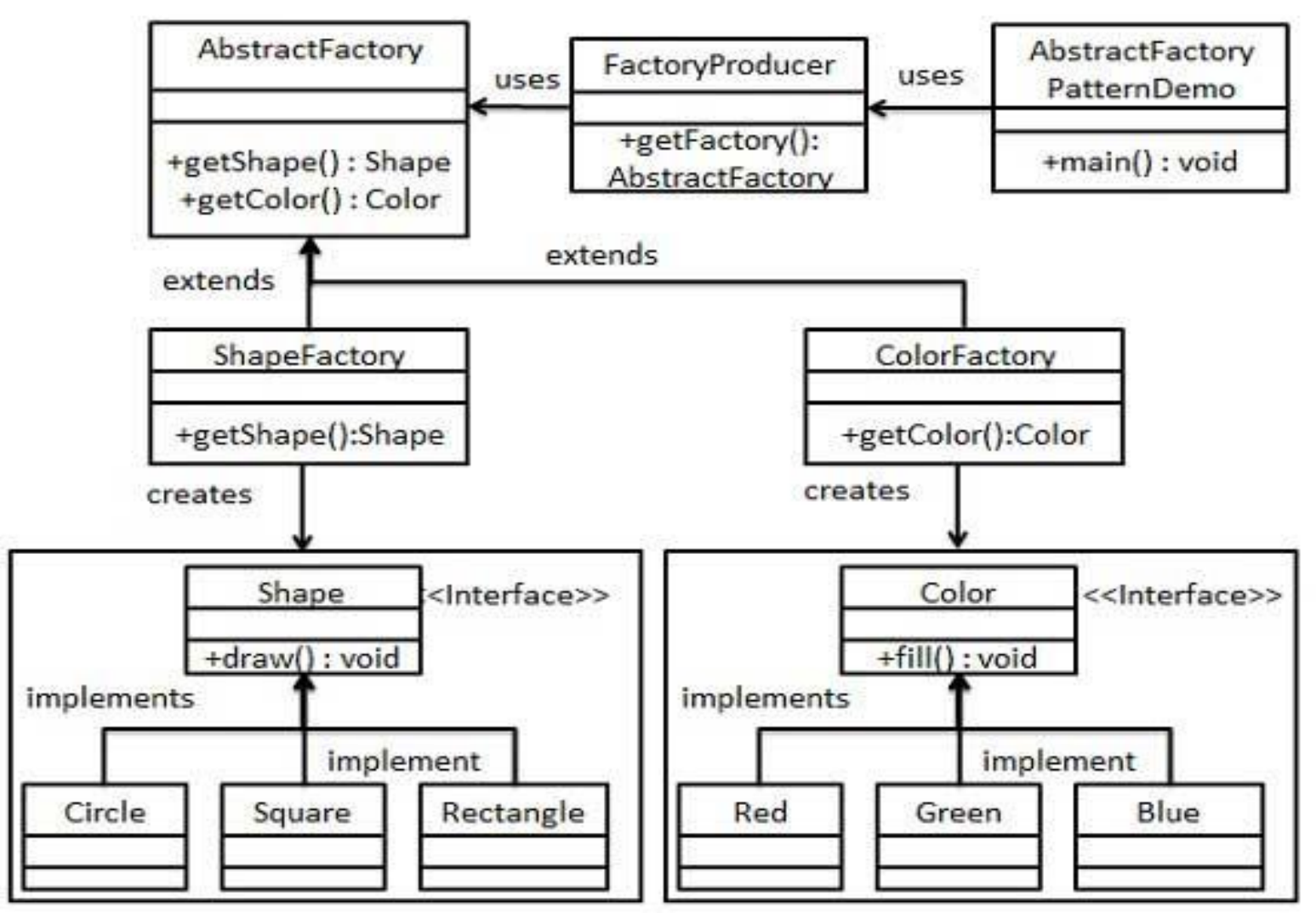
- **Método de fábrica** se utiliza para crear un solo producto, y **Abstract Factory** se trata de crear familias de productos relacionados o dependientes.
- Patrón **Factory Method** expone un método para el cliente para crear el objeto, mientras que en caso de **Abstract Factory** se exponen una familia de objetos relacionados que pueden consistir de estos métodos de fábrica.
- Patrón **Factory Method** oculta la construcción de solo un objeto donde como **método de fábrica Abstract** oculta la construcción de una familia de objetos relacionados. Las Fábricas abstractas se implementan normalmente utilizando (un conjunto de métodos de fábrica).

- Patrón **AbstractFactory** utiliza la composición de delegar la responsabilidad de crear objetos a otra clase, mientras que el patrón de diseño **de fábrica** utiliza la herencia y se basa en la clase derivada o subclase para crear objetos.
- La idea detrás del patrón **Factory Method** es que permite para el caso de que un cliente no sabe qué clases de concreto será necesario para crear en tiempo de ejecución, pero sólo quiere una clase que hará el trabajo, mientras que el patrón **AbstractFactory** es mejor utilizada cuando el sistema tiene que crear varias familias de productos o si desea proporcionar una biblioteca de productos sin exponer a los detalles de implementación.

IMPLEMENTACIÓN CON EL FACTORY METHOD



IMPLEMENTACIÓN CON EL ABSTRACT FACTORY



FIN