DESIGN PATTERNS

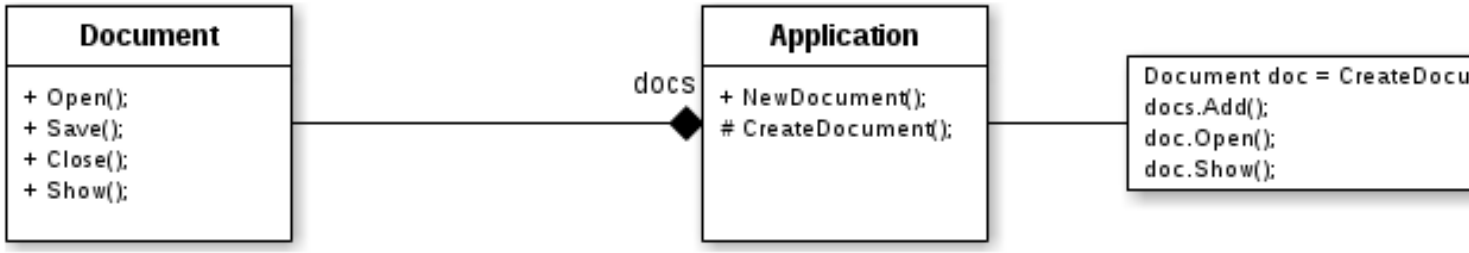
FACTORY METHOD

INTENCIÓN

Factory Method es un patrón de diseño, que se encuentra dentro de la clase de los patrones creacionales. Define una interfaz para crear un objeto, permitiendo a las subclases decidir que clases instancia, es decir, permite a una clase derivar la instanciación a las subclases.

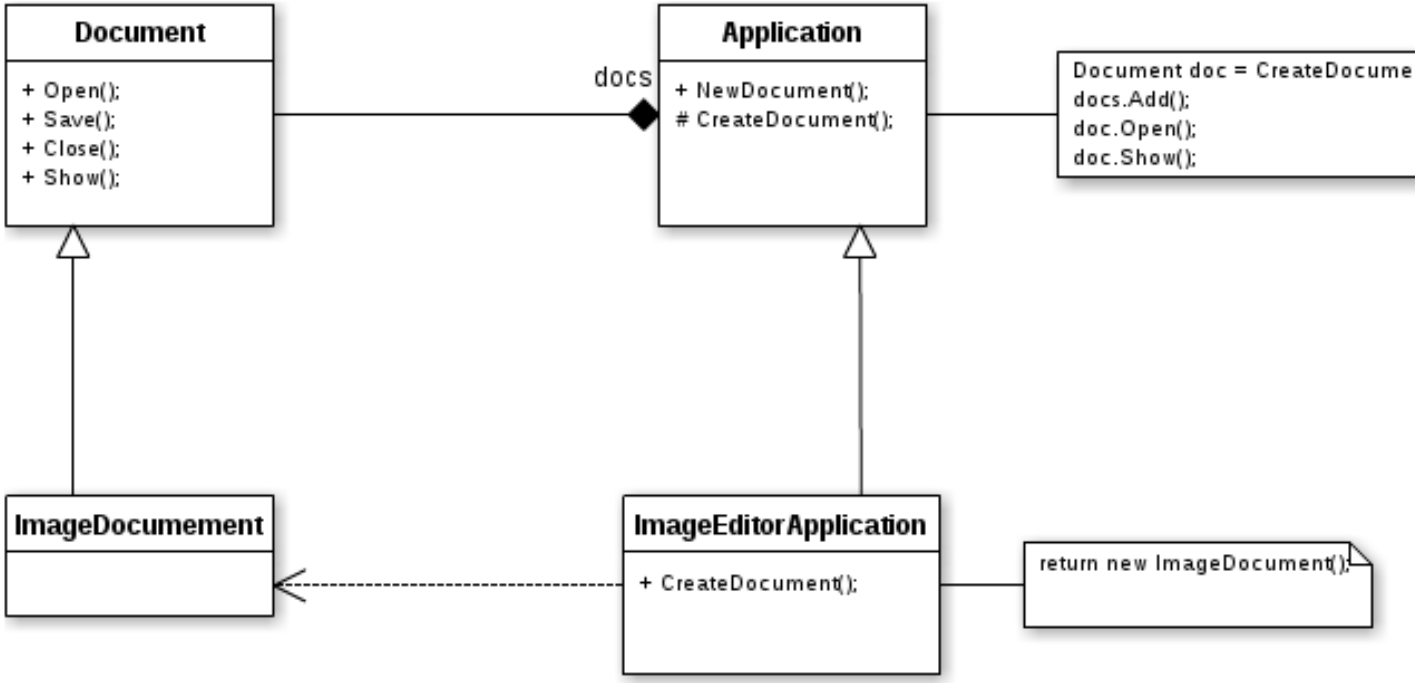
MOTIVACIÓN

onsideremos un **framework** ara aplicaciones que ueden presentar múltiples ocumentos al usuario. Dos **ostracciones** claves en este ntorno son las clases pplication y Document. mbas son clases abstractas, s clientes deben crear sus spectivas subclases para enerar las nplementaciones pecíficas.



MOTIVACIÓN

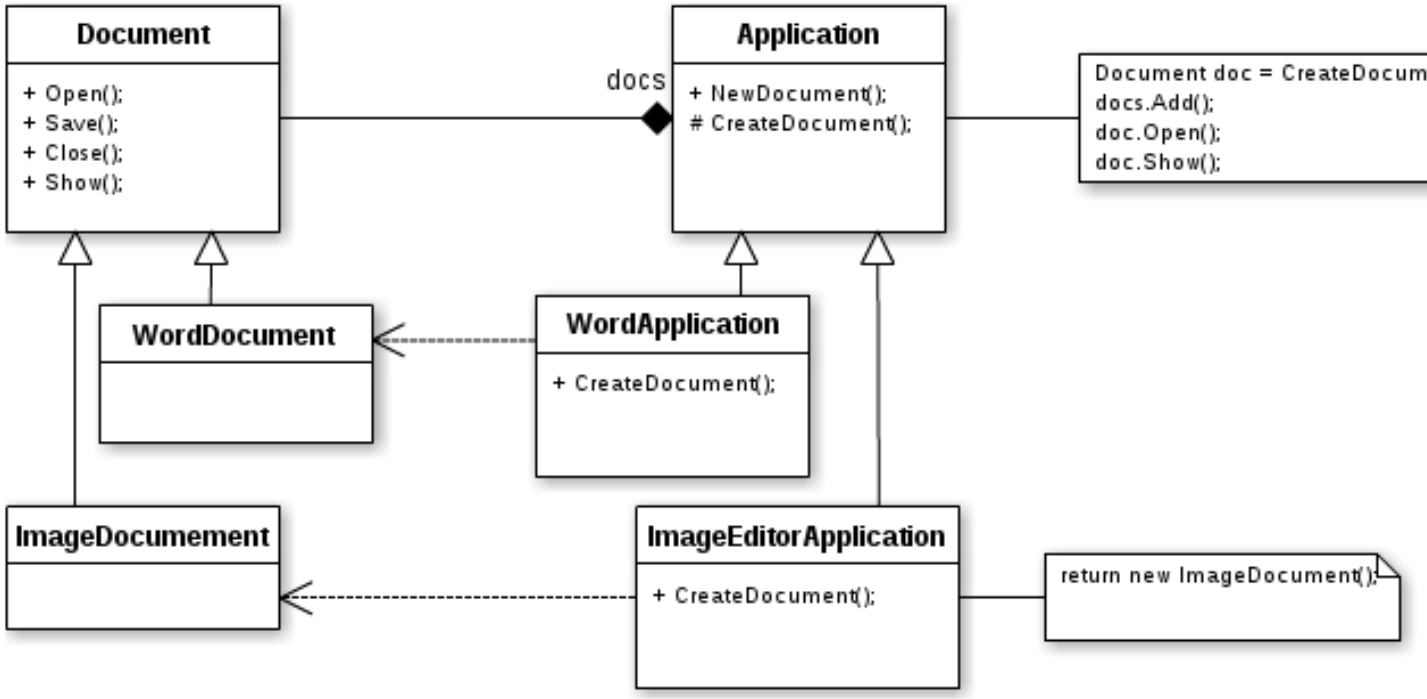
ara crear una aplicación de ditor de imágenes, efiniremos las clases oncretas nageEditorApplication y nageDocument. La clase pplication es la responsable e controlar a Document y s creará como sea querido por el usuario ramework), por ejemplo, uando el usuario seleccione pen o new, desde un menú.



SOLUCIÓN

clase Application no
uede predecir que subclase
e Document instanciar,
onoce cuando un nuevo
ocumento debería ser
eado, pero no sabe que
ocumento concreto.

ediante el Factory Method
obtiene la solución, éste
ncapsula el conocimiento
e la subclase Document a
ear, y Application hace uso
e ello sin tener el
onocimiento de la
nplementación del mismo.



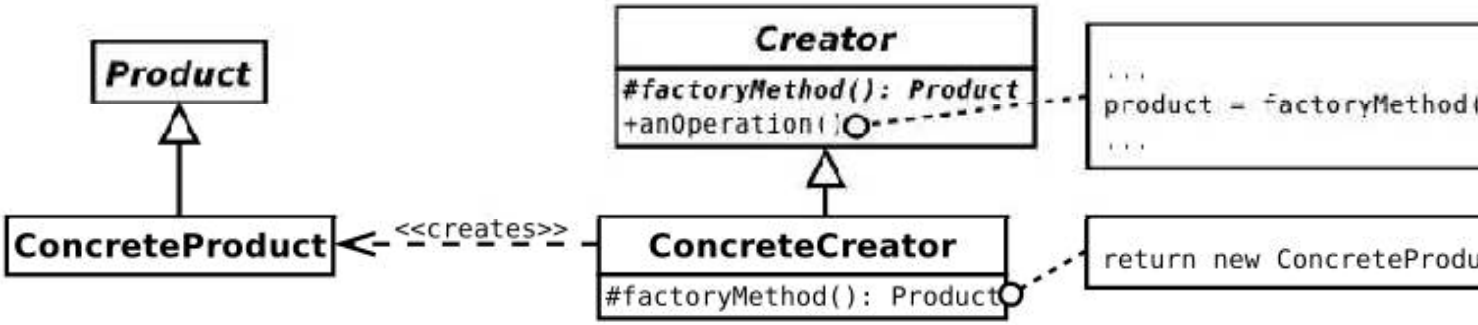
PARTICIPANT ES

oduct, define la interfaz que odela un objeto.

oncreteProduct, implementa oduct.

eator, declara el Factory ethod.

oncreteCreator, implementa al eador abstracto (Creator).



APLICACIONES

Usaremos Factory Method cuando:

- Una clase no puede anticipar la clase de objeto que debe crear.
- Se desea que las subclases especifiquen el objeto que se crea.
- Una clase delega la responsabilidad a una de las subclases auxiliares y se quiere saber cual es la subclase delegada.

CONSECUENCIAS

+ Se gana en flexibilidad, pues crear los objetos dentro de una clase con un “Factory Method" es siempre más flexible que hacerlo directamente, debido a que se elimina la necesidad de atar nuestra aplicación a clases de productos concretos.

+ Elimina la necesidad de incluir clases especificas de la aplicación en códigos más generales.

+ Se facilita, en cuanto a que se hace natural, la conexión entre jerarquías de clases paralelas, que son aquellas que se generan cuando una clase delega algunas de sus responsabilidades en una clase aparte. Ambas jerarquías de clases paralelas son creadas por un mismo cliente y el patrón Método de Fábrica establece la relación entre parejas de subclases concretas en las mismas.

- Se obliga al cliente a definir subclases de la clase "Creador" sólo para crear un producto concreto y esto puede no ser apropiado siempre.

BIBLIOGRAFÍA

- Design Patterns: Elements of Reusable Object-Oriented Software