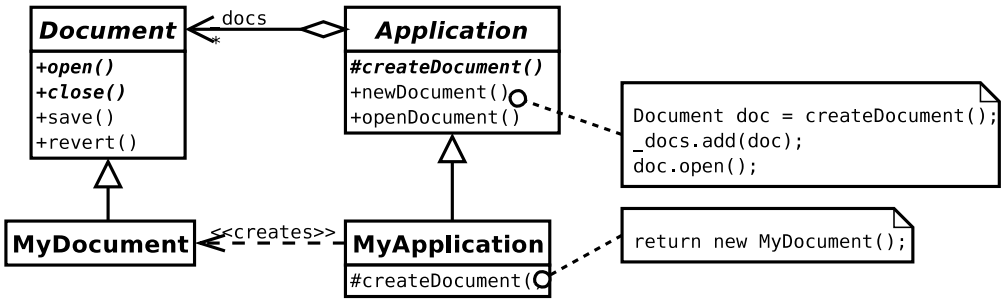
Método de Fabricación (Factory Method)

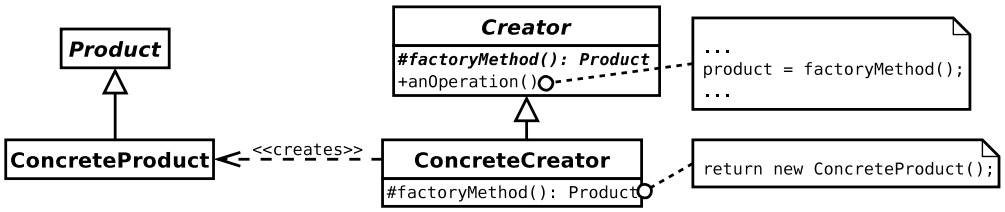
- *Patrón de Creación*
- *Propósito*

  Define interfaz para crear un objeto, pero deja que las subclases decidan que clase concreta instanciar

- *Motivación*
  - *Framework* para aplicaciones que permiten presentar múltiples documentos al usuario
  - Dos abstracciones: Aplicación y Documento
  - Para una aplicación específica, se especializan los conceptos de Aplicación y Documento
  - Problema: La abstracción Aplicación conoce *cuándo* se debe crear un documento, pero no *qué* documento concreto
  - Solución: Asigna a un método abstracto la responsabilidad de crear un documento concreto, separando esta responsabilidad del framework.



- *Aplicabilidad*
  - No se puede anticipar la clase de objetos a crear
  - Una clase quiere que sus subclases especifiquen que objetos se deben crear
  - clases delegan la responsabilidad a una de muchas subclases colaboradoras, y se desea localizar el conocimiento de qué subclase es la encargada
- *Estructura*



- *Participantes*
  - Producto (Product)

    Define la interfaz de los objetos creados por el método de fabricación
  - Producto Concreto (ConcreteProduct)

    Implementa la interfaz de los productos
  - Creador (Creator)

    Declara el método de fabricación

    Opcionalmente, el creador puede definir una implementación por defecto que construye un producto concreto

    Puede utilizar el método de fabricación

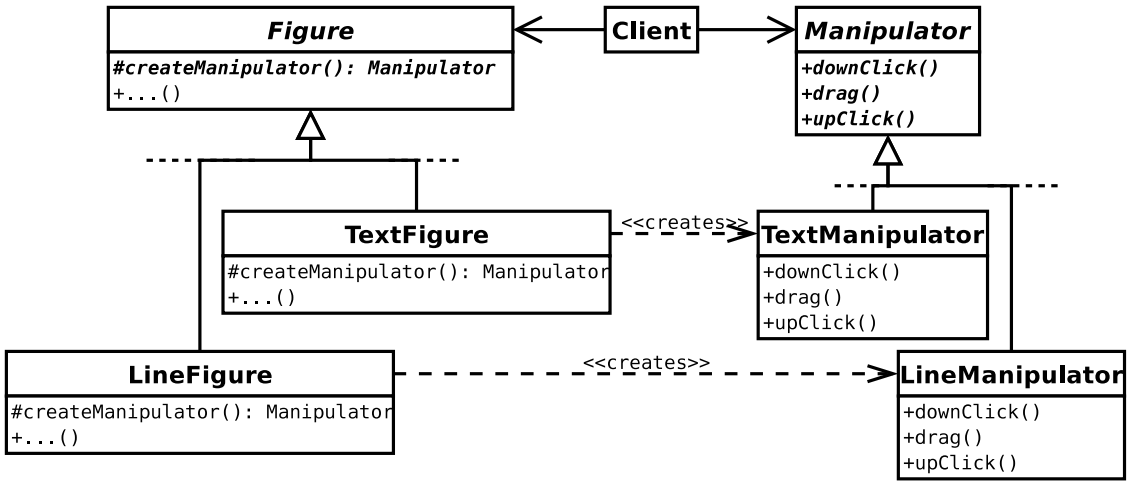
- *Participantes* (cont.)
  - Creador concreto (ConcreteCreator)

    Redefine el método de fabricación para devolver una instancia de un producto concreto

- *Colaboraciones*

  El creador emplea el método de fabricación redefinido por sus subclases para utilizar la instancia de producto concreto apropiada

- *Consecuencias*
  - Elimina la necesidad de incluir clases específicas de la aplicación en código más general (potencia reutilización del *framework*)
  - Mayor flexibilidad dado que se proporciona un mecanismo a las subclases para introducir una versión extendida del producto
  - Conecta jerarquías paralelas



- *Consecuencias* (cont.)
  - Desventaja: Puede obligar a extender la clase creadora sólo para crear un producto concreto
    - si esto es un problema se puede emplear otra solución (por ejemplo, Prototipo)

- *Implementación*
  - Dos variantes:
    - El creador es una clase abstracta y no proporciona implementación por defecto para el método (necesario extender al creador)
    - El creador es una clase concreta y define un producto concreto por defecto (flexibilidad para cambios futuros)
  - Métodos de fabricación parametrizados
    - un único método puede crear distintos productos en base a los parámetros del método de fabricación