**FACTORY METHOD PATTERN**

**LOS PRINCIPIOS DE DISEÑO UTILIZADOS EN ESTE PATRÓN SON:**

* El principio abierto/Cerrado
* Principio de responsabilidad única
* Depende de abstracciones, no de implementaciones

**LOS POSIBLES CASOS DE USO DE ESTE PATRÓN SERIAN:**

* Cuando no sepas los tipos exactos y/o dependencias con las que los objetos de tu código deberían trabajar.
* Cuando desees proporcionar a los usuarios de tu biblioteca o marco una forma de ampliar sus componentes internos.
* Cuando desees ahorrar recursos del sistema reutilizando objetos existentes en lugar de reconstruirlos cada vez.

**BENEFICIOS:**

* Permite aislar clases concretas: ayuda a controlar las clases de objetos que se crean en la aplicación.
* Encapsula la responsabilidad y el proceso de creación de objetos producto.
* El sistema estará adaptado para recibir futuras ampliaciones
* Reduce el acoplamiento de los productos con sus creadores.

**DESVENTAJAS:**

* El código puede volverse más complicado ya que necesita introducir muchas subclases nuevas para implementar el patrón.

**Guías de diseño del patrón “Depende de abstracciones, no de implementaciones”:**

* Ninguna variable debe contener una referencia a una clase concreta.
* Ninguna clase debe derivar de una clase concreta.
* Ningún método debe anular un método implementado de ninguna de sus clases base.

Estas guías no son forzosas, ya que si las seguimos a pie de la letra nunca terminaríamos nuestro programa, pero seguirlas te ayudara a pensar cual es el nivel de abstracción correcto y el punto adecuado para romper estas guías.