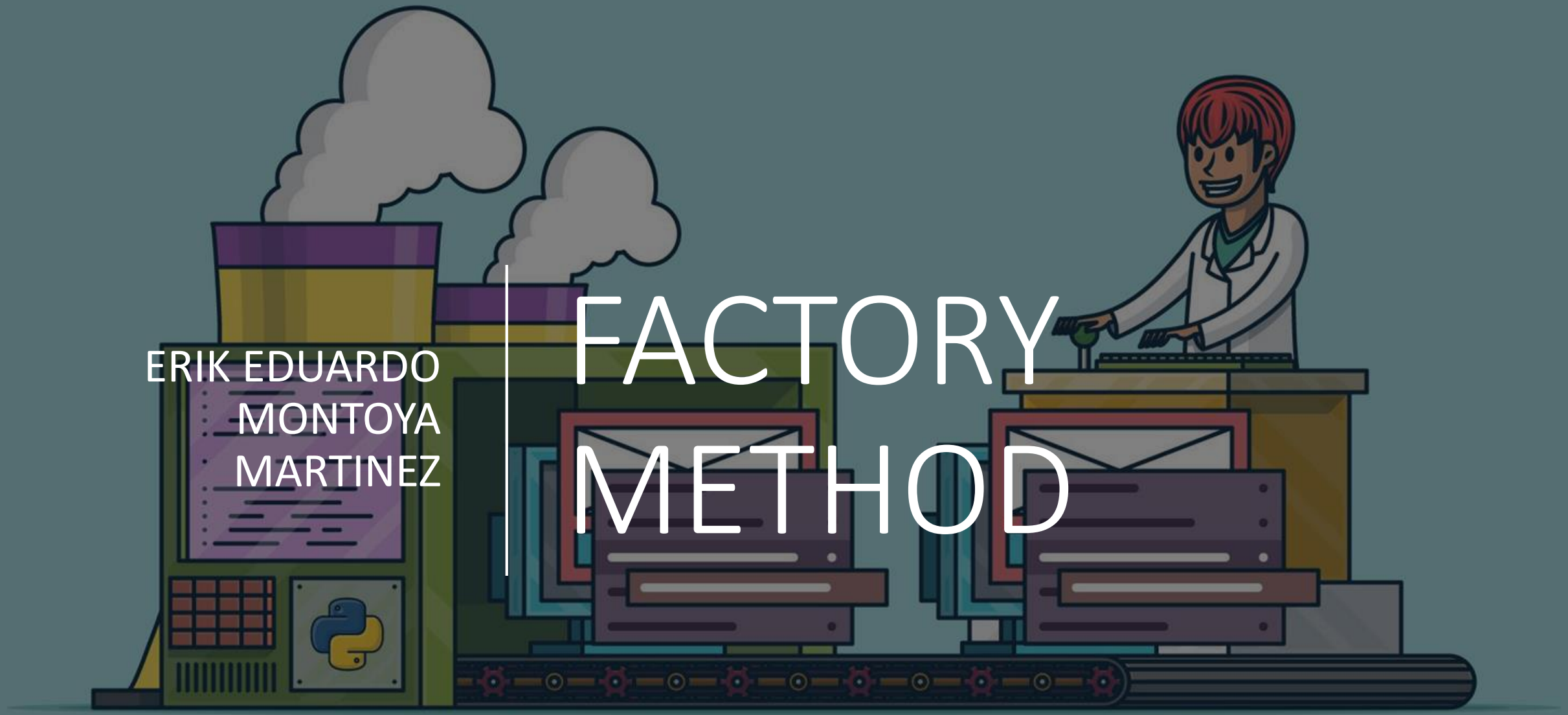


FACTORY METHOD

ERIK EDUARDO MONTOYA MARTINEZ

Real Python

PATRÓN CREACIONAL

- Es un patrón de diseño que define una interfaz para crear un objeto
- Permite escribir aplicaciones que son más flexibles respecto de los tipos a utilizar difiriendo la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema.
- Permite también encapsular el conocimiento referente a la creación de objetos.
- Hace que el diseño sea mas personalizado y solo un poco mas complicado.

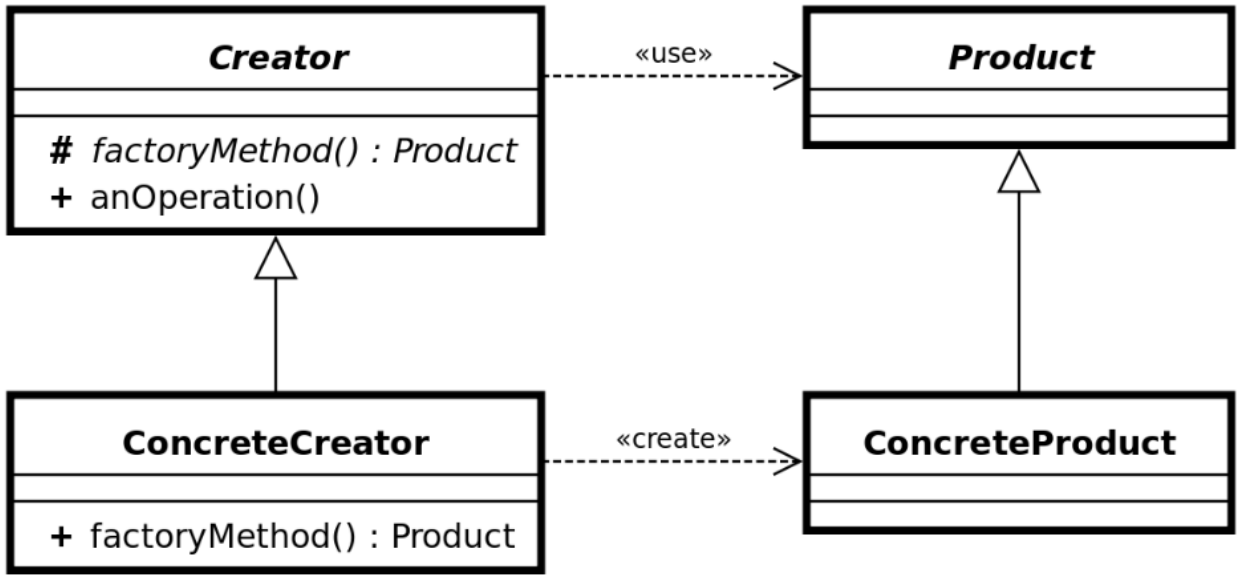
Propósito

- Definir una interfaz para la creación de un objeto, pero permitiendo a las subclases decidir de que clase instanciarlo. Permite, por tanto, que una clase difiera la instanciación en favor de sus subclases.

SOLUCION

El método Factory sirve para crear objetos como el Template para implementar algoritmos.
Una super clase especifica todos los comportamientos estándar y genéricos, y luego delega los detalles de creación a las subclases proporcionadas por el cliente.
Otros patrones de diseño requieren nuevas clases, mientras que este método solo requiere una nueva operación.

- **Producto:** Define la interfaz de los objetos que crea el método de fabricación.
- **ProductoConcreto:** Implementa la interfaz Producto.
- **Creador:** Declara el método de fabricación, el cual devuelve un objeto del tipo Producto. También puede definir una implementación predeterminada del método de fabricación que devuelve un objeto
- **CreadorConcreto:** Redefine el método de fabricación para devolver una instancia de ProductoConcreto.



Debe usarse:

- Cuando una clase no puede adelantar las clases de objetos que debe crear.
- Cuando una clase pretende que sus subclases especifiquen los objetos que ella crea.
- Cuando una clase delega su responsabilidad hacia una de entre varias subclases auxiliares y queremos tener localizada a la subclase delegada.

Ventajas

- Se gana en flexibilidad, pues crear los objetos dentro de una clase con un "Método de Fábrica" es siempre más flexible que hacerlo directamente, debido a que se elimina la necesidad de atar nuestra aplicación a unas clases de productos concretos.
- Se facilitan futuras ampliaciones, puesto que se ofrece las subclases la posibilidad de proporcionar una versión extendida de un objeto, con sólo aplicar en los Productos la misma idea del "Método de Fábrica".
- 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.

Inconvenientes

- Se obliga al cliente a definir subclases de la clase "Creador" sólo para crear un producto concreto y esto puede no ser apropiado siempre.
- Vuelve al código mas difícil de leer ya que todo tu codigo esta detras de una abstraccion que a su vez puede ocultar abstracciones.
- Puede clasificarse como un anti-patron cuando no se usa correctamente.
- Puede presentar este patrón es que puede requerir crear una nueva clase simplemente para cambiar la clase de Producto.

Consecuencias

- Elimina la necesidad de introducir clases específicas en el código del creador.
- Facilita el control de las extensiones de las subclases. Se puede ofrecer una implementación por defecto para el método fábrica (en este caso no sería abstracto) y redefinirlo en el método fábrica de la fábrica concreta.
- Los métodos fábricas pueden verse como métodos de enganche.

Relacionado con

- Fábrica Abstracta es frecuentemente implementado con el Método de Fábrica.
- Método de Fábrica habitualmente se utiliza dentro del Método Plantilla.
- Se utiliza frecuentemente con Template Method.
- Otros nombres: Virtual Contructor (Constructor virtual)

Ejemplo de aplicación

- **En un "marco de referencia" para aplicaciones que pueden presentar múltiples documentos al usuario, no se sabe, a priori, los tipos de documentos con los que va a trabajar cada aplicación concreta. El "marco de referencia" debe, por tanto, instanciar clases, pero sólo tiene conocimiento de las clases abstractas, las cuales no pueden ser instanciadas.**
- **La solución está en hacer que CrearDocumento() sea un Método de Fábrica que se encargue de la "fabricación" del objeto oportuno en cada situación (es una operación abstracta en la clase abstracta, pero concreta y distinta en las subclases correspondientes a cada tipo de aplicación.**