UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y SISTEMAS

Métodos Factory

CURSO:

- JAVA PROGRAMACIÓN

ALUMNOS :

- ASTO RUPAY JOAQUÍN RODRIGO
- MURRIETA MEZA DALE ALEJANDRO

PROFESOR:

- GUSTAVO CORONEL

2020

Dedicamos este trabajo a nuestro docente Gustavo Coronel por haber brindado sus conocimientos para el desarrollo de este trabajo

Índice

Resumen

Los patrones de fabricación son aquellos patrones que implican algún tipo de factoría de objetos. Los objetos de fabricación tienen la responsabilidad de crear instancias de objetos de otras clases. Además, son responsables de encapsular la forma en que se crean tipos específicos de objetos en una aplicación. Existen tres tipos de patrones de fabricación: ***Factory Method***, define una interfaz para crear objetos, pero deja que sean las subclases las que deciden qué clases instanciar; ***Abstract Factory***, Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas; y ***Simple Factory***, Clase utilizada para crear nuevas instancias de objeto. En el transcurso del informe profundizaremos más respecto a estos tres tipos.

Desarrollo teórico

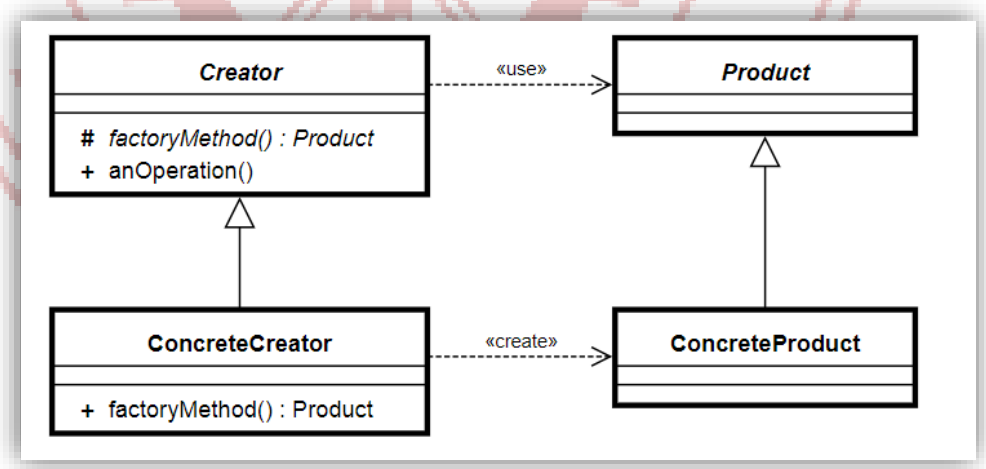
- Conceptos previos

  - ✓ ***Patrones de diseño***: Son técnicas para resolver problemas comunes en el desarrollo de *software* e interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

  - ✓ ***Fábrica***: También llamada factoría o Factory (en inglés). Es una clase que implementa uno o más métodos de creación, que son los métodos que se encargan de crear instancias de objetos (estas instancias pueden ser de esta misma clase o de otras). Esta clase tiene entre sus responsabilidades la creación de instancias de objetos, pero puede tener también otras responsabilidades adicionales. Los métodos de creación pueden ser estáticos.

- Tipos de patrones de fabricación

  - ✓ ***Factory Method:*** Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue a sus subclases la creación de objetos. Diagrama:

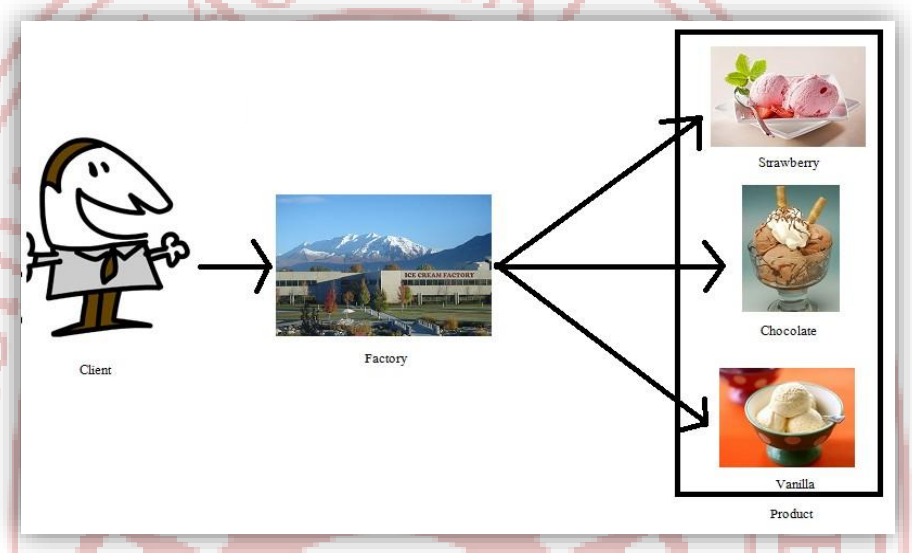


Participantes:

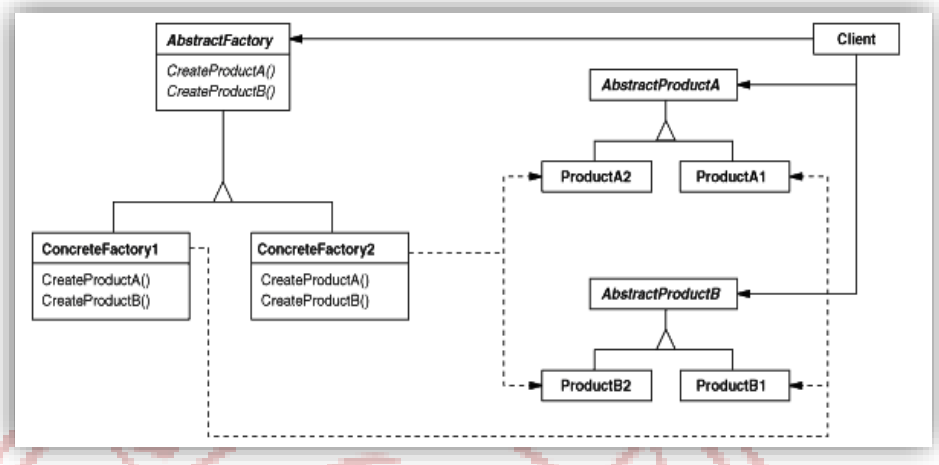
- ***Producto:*** Define la interfaz de los objetos que crea el método de fabricación.

- ***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 ProductoConcreto. Puede llamar al método de fabricación para crear un objeto Producto.
- ***CreadorConcreto:*** Redefine el método de fabricación para devolver una instancia de ProductoConcreto.

Ejemplo en la vida real del Factory Method:



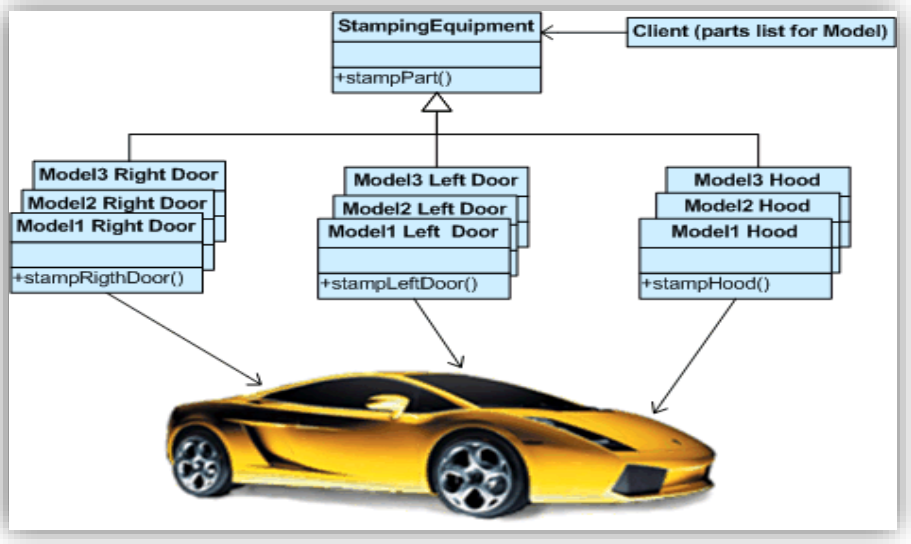
- ✓ ***Abstract Factory:*** Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas. Abstract Factory puede ser utilizado para desarrollar frameworks y sistemas que pueden ser configurados con una de múltiples familias de productos. Abstract Factory generalmente se implementa utilizando Factory Method y por tanto provee al menos toda la flexibilidad de éste. La diferencia principal entre ambos es que el primero trata con familias de productos, mientras que el otro se preocupa por un único producto.
  Diagrama:

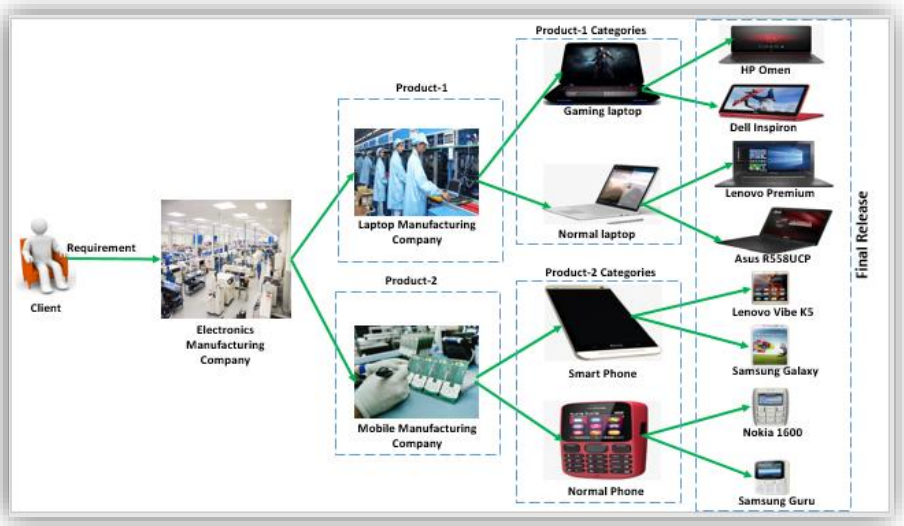


Participante:

- ***FabricaAbstracta:*** Declara una interfaz para operaciones que crean objetos producto abstractos.
- ***FabricaConcreta:*** Implementa las operaciones para crear objetos producto concretos.
- ***ProductoAbstracto:*** Declara una interfaz para un tipo de objeto producto.
- ***ProductoConcreto:*** Define un objeto producto para que sea creado por la fábrica correspondiente. Implementa la interface ProductoAbstracto.
- ***Cliente:*** Sólo usa interfaces declaradas por las clases FabricaAbstracta y ProductoAbstracto.

Ejemplos en la vida real del Abstract Factory:





- ✓ ***Simple Factory***: Clase con la responsabilidad de crear objetos de otras clases. No delega en subclases y sus métodos pueden ser estáticos. Puede evolucionar a un Factory Method o Abstract Factory. Hay casos que no son cubiertos por Factory Method o Abstract Method como por ejemplo, clases con métodos estáticos de fabricación o fábricas concretas que tienen implementación, pero sus métodos no son redefinibles. En estos casos, estamos ante una implementación del patrón **Simple Factory**.

Soluciones

-**Factory Method:**

Las clases principales en este patrón son el creador y el producto. El creador necesita crear instancias de productos, pero los tipos concretos de cada producto no deben estar reflejadas en el propio creador sino que las posibles subclases del creador deben poder especificar los tipos concretos, subclases, de los productos para utilizar.

La solución para esto es hacer un método abstracto (el método de la fábrica) que se define en el creador. Este método abstracto se define para que devuelva un producto. Las subclases del creador pueden sobrescribir este método para devolver subclases apropiadas del producto

-**Abstract Factory**:

Se solicita la creación de diferentes vehículos de transporte (Buses, Busetas y Taxis) sin que se especifique en detalle la forma de su creación.

La solución es utilizar el patrón Abstract Factory para independizar la forma como crearemos los objetos, de esta manera creamos familias de objetos de tipo Vehículo delegando el proceso y sin tener que entrar en detalles desde la clase solicitante.

-**Factory Pattern (Simple Factory):**

Normalmente al ver al Factory Pattern (Simple Factory) se diría que se refiere a uno de los dos patrones que hemos estudiado anteriormente. Sin embargo, hay casos que no son cubiertos por estos patrones por lo que el Factory Pattern (Simple Factory) brinda soluciones alternativas como por ejemplo, clases con métodos estáticos de fabricación o fábricas concretas que tienen implementación, pero sus métodos no son redefinibles. En estos casos, estamos ante una implementación del patrón Simple Factory.

Conclusiones

-El **patrón Factory Method** 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. Factory Method hace también que el diseño sea más adaptable a cambio de sólo un poco más de complejidad.

-**El patrón Abstract** Factory es un Patrón que al principio puede sonar un poco intimidante pero a medida que vamos trabajando con el nos damos cuenta de cómo podemos sacarle provecho a su aplicación.

Podemos evidenciar también el uso de varios conceptos de programación orientada a objetos siendo los patrones de diseño un gran ejemplo de su aplicación

-**En el patrón Simple Factory Pattern**, su principio fundamental es crear una abstracción que decida cuál de las varias clases posibles devolver. Un desarrollador simplemente llama a un método de la clase sin conocer los detalles de implementación o qué subclase realmente utiliza para implementar la lógica.

Este enfoque ayuda a mantener los problemas de dependencia de datos separados de los métodos útiles de las clases.

Recomendaciones

Se recomienda la aplicación de los patrones Factory para buscar siempre un sistema más óptimo y con una arquitectura definida, cual o cuales de los tres patrones, presentados en este informe, se debe aplicar pues eso depende principalmente de las necesidades del sistema.

Ala vez se puede seguir la recomendación de Robert Martin (ingeniero de software y autor estadounidense) al momento de utilizar estos **Patrones de Fabricación**. Donde menciona: "*No utilizar por defecto, y no comenzar utilizándolas frente al primer indicio de que puedan serle útiles... Aguarde a tener un problema concreto que las fábricas puedan resolver. Y en ese caso, no dude en utilizarlas*".