2015

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

CURSO: LENGUAJE DE PORGRAMACIÓN ORIENTADO A OBJETOS

PROFESOR: HANCCO CARPIO, RONY JORDAN

CICLO: 2015-I

ESTUDIANTE: CALACHUA RAMOS, GERSON ALONSO 20134549ª

TRABAJO: ENSAYO N°7

Alumno
[Escriba el nombre de la compañía]
01/01/2015

SEVENTH ESSAY

En este séptimo desarrollaremos un único tema en particular el cual tuvo un breve desarrollo en la anterior clase de nuestro curso de LPOO, el tema a desarrollar será el denominado “Abstract Factory”.

Para comenzar debemos aclarar que el Abstract Factory es en realidad un patrón de diseño para el desarrollo de software, y ahora nos hacemos la pregunta ¿Qué es un patrón de diseño? Es en este caso donde nos vemos en la necesidad de aclarar nuestras dudas y explicar primeramente que es un patrón de diseño para luego poder profundizar en el tema de fondo que es el Abstract Factory.

Los patrones de diseño son la base para la solución de problemas recurrentes que se nos puedan presentar a la hora de implementar o programar una aplicación, en pocas palabras resulta ser una solución a un problema de diseño, para que un patrón de diseño sea considerado como tal debe de poseer ciertas características entre las cuales destaca su efectividad (poder resolver problemas similares en diferentes ocasiones) y su capacidad de ser reutilizable.

De entre las ventajas que nos proporciona el uso de los patrones de diseño tenemos.

- Evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formaliza un vocabulario común entre diseñadores.
- Estandariza el modo en que se realiza el diseño.

Si bien es cierto que estos patrones se van actualizando y mejorando con el tiempo y cada vez existen más, para ayudarnos en su utilización estos se han clasificado en tres grupos.

A. PATRONES CREACIONALES:

   Son utilizados para instanciar objetos, busca separar la implementación del cliente de la de los objetos que se emplean, con ello se procura independizar al sistema de como sus objetos son creados y/o representados.

B. PATRONES ESTRUCTURALES:

   Esto afecta a la manera en que los objetos se conectan con otros objetos, son utilizados para organizar clases u objetos para conformar estructuras más complejas.

C. PATRONES COMPORTACIONALES:

   Se utilizan para definir como las clases y los objetos interaccionan entre ellos.

La imagen que se muestra a continuación busca ilustrar más lo ya mencionado y darnos a conocer algunos de los patrones de diseño más conocido y la clasificación en la que se encuentran.

Patrones de diseño				
		Propósito		
		De Creación	Estructurales	De Comportamiento
Ámbito	Clase	• Factory Method	• Adapter (de clases)	• Interpreter • Template Method
	Objeto	• Abstract Factory • Builder • Prototype • Singleton	• Adapter (de objetos) • Bridge • Composite • Decorator • Façade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

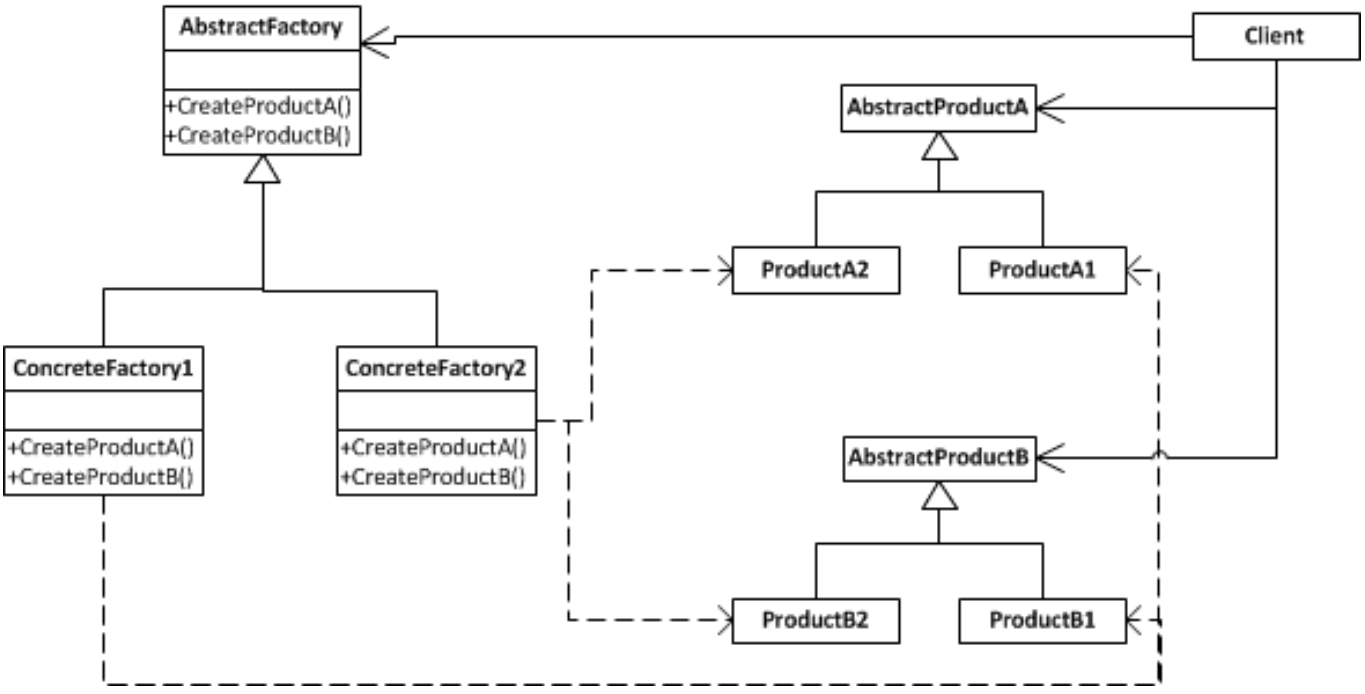
Luego de haber ampliado nuestro conocimiento acerca de lo que es y para que sirven los patrones de diseño estamos listos para hablar acerca de nuestro tema de fondo que es el Abstract Factory.

Como ya mencionamos, el Abstract Factory (Fábrica abstracta) es un patrón de diseño para el desarrollo de software y si se tomó la molestia de revisar la imagen de arriba se dará cuenta que está clasificado como un patrón del tipo creacional.

Este patrón nos permite crear mediante una interfaz *"conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cuál es el objeto concreto."*

Este patrón se puede usar por ejemplo si queremos crear diferentes objetos los cuales pertenecen a una misma familia, el problema que buscará solucionar este patrón es el de crear diferentes familias de objetos brindándonos flexibilidad a la hora de aislar a las clases concretas y facilitar los cambios en las familias de los objetos, no obstante como cualquier patrón tiene sus por y contras y un defecto que se puede mencionar es que para agregar nuevos objetos se deben modificar tanto las fábricas abstractas como las concretas.

Para entender mejor esta última definición nos valdremos de la siguiente imagen y una pequeña explicación de la estructura que presenta.



- **Client:** Sólo usa interfaces declaradas por las clases Abstract Factory y Abstract Product.
- **Abstract Factory:** Declara una interfaz para operaciones que crean objetos producto abstractos.
- **Concrete Factory:** Implementa las operaciones para crear objetos producto concretos.
- **Abstract Product:** Declara una interfaz para un tipo de objeto producto.
- **Product:** Define un objeto producto para que sea creado por la fábrica correspondiente. Implementa una interfaz producto.

De esta manera finalizamos el presente ensayo esperando que al lector le haya sido de ayuda la información aquí brindada.

BIBLIOGRAFÍA

- http://es.wikipedia.org/wiki/Abstract_Factory
- http://lineadecodigo.com/categoria/patrones/
- http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o
- http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos
- http://es.slideshare.net/ikercanarias/patrones-de-diseo-de-software-14836338
- http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol2/ch09s02.html
- http://es.slideshare.net/zykharyo/abstract-factory-10276102