“UNIVERSIDAD NACIONAL DE INGENIERIA”

“Facultad de Ingeniería Industrial y de Sistemas”

Alumno: Vidal Flores, Jose Carlos

20130078D

Dirigido: HANCCO CARPIO, RONY JORDAN

Curso: LENGUAJES DE PROGRAMACIÓN ORIENTADOS A OBJETOS (ST-232 V)

“2015-I”

Interfaces

Las interfaces en java nos permiten declarar las especificaciones necesarias para una clase o un conjunto de clases que necesiten de cierto tipo de requerimientos en común.

Las interfaces poseen la característica de herencia múltiple, en el cual no sucedía con una clase, donde cierta clase solo podía heredar de una clase, la interface en cambio puede heredar de una lista de interfaces.

Una interfaz puede ser declarada de la siguiente manera:

> modificadordeinterfaz *interface* NombreInterface{
> declaración de métodos de la interfaz
> declaración de variables de la interfaz }

En la interface solo se declara las clases, y algunas variables estáticas con valores definidos, mas no se especifica ni se implementa las acciones de los métodos; se identifican con la palabra reservada “interface” antes del nombre.

Para implementar una interfaz en una clase, se usa la siguiente sintaxis:

> Modificadores class nombredeclase implements
> listadenombresdeinterfaces {
> declaraciones }

Una clase puede implementar una lista de interfaces separados por comas, y la implementación viene dada por la palabra reservada “implements”.

Las clases que implementan una interfaz deben implementar todos los métodos declarados en dicha interface, estas clases son clases concretas; además se puede definir clases como abstractas que pueden o no implementar todos los métodos de la interface implementada.

Abstract Factory

El patrón Abstract Factory 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.

En este patrón de diseño se debe crear una interfaz o clase abstracta (usualmente denominada abstractFactory) que permita especificar los métodos que debe tener las clases concretas llamadas fabricas; que sirve para la instanciación de objetos de las clases productos relacionadas, con el fin de no hacer uso directamente del nombre de la clase producto a desarrollar; además

se debe crear una interfaz para cada tipo de producto en concreto que permita especificar los métodos que esos requieren.

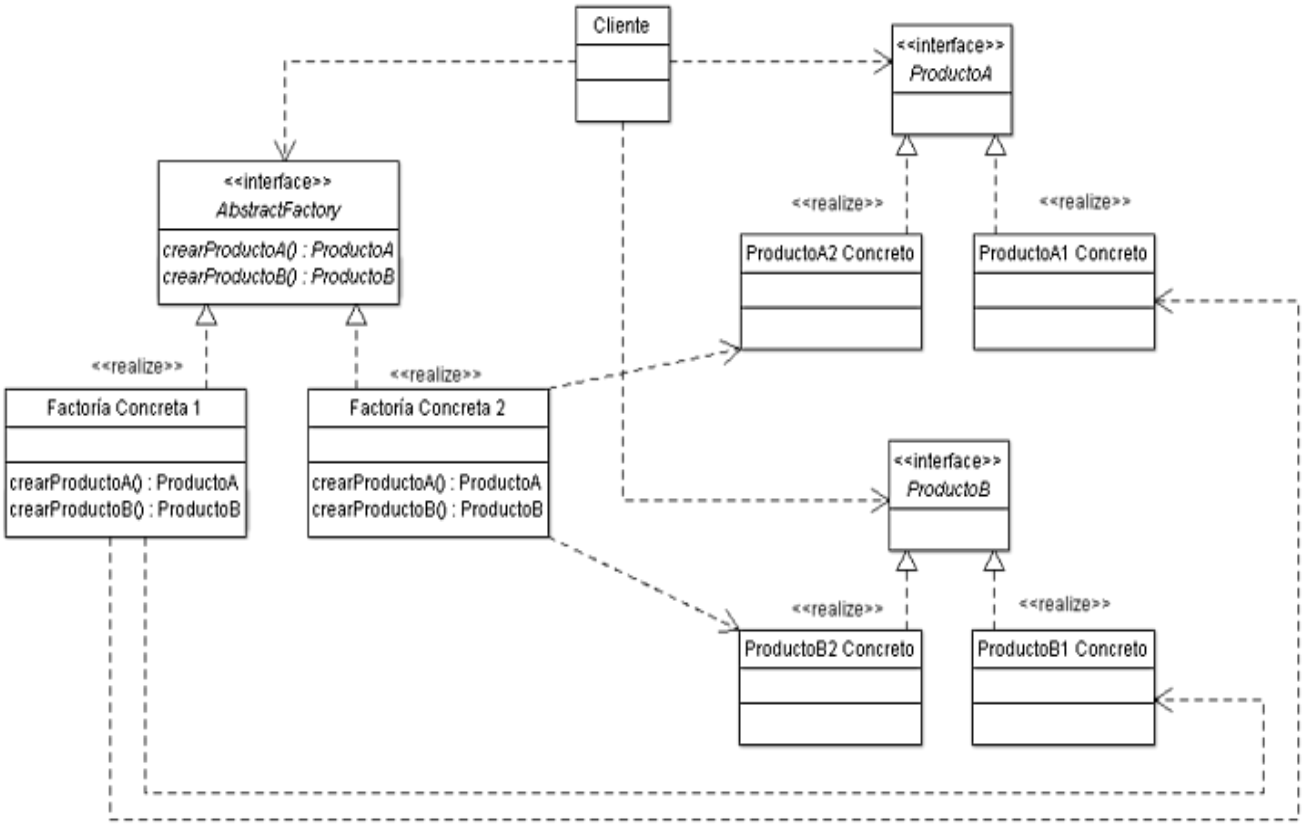
Este patrón de diseño suele ser muy útil gracias a que ayuda en el mantenimiento del sistema creado, además de favorecer al encapsulamiento.

Usos del patrón Abstract Factory

Este patrón se puede aplicar cuando:

- Un sistema debe ser independiente de cómo sus objetos son creados.
- Un sistema debe ser configurado con una cierta familia de productos.
- Se necesita reforzar la noción de dependencia mutua entre ciertos objetos.

Estructura del Patrón Abstract Factory



La estructura típica del patrón Abstract Factory es la siguiente:

- ✓ **Cliente:** La clase que llamará a la factoría adecuada ya que necesita crear uno de los objetos que provee la factoría, es decir, Cliente lo que quiere es obtener una instancia de alguno de los productos (ProductoA, ProductoB).

- ✓ **AbstractFactory:** Es la definición de la interfaces de las factorías o fábricas. Debe de proveer un método para la obtención de cada objeto que pueda crear. ("crearProductoA()" y "crearProductoB()").

- ✓ **Factorías Concretas:** O también llamadas fabricas concretas, estas son las diferentes familias de productos. Provee de la instancia concreta de la que se encarga de crear.

- ✓ **Producto abstracto:** Definición de las interfaces para la familia de productos genéricos. En el diagrama son "ProductoA" y "ProductoB".

- ✓ **Producto concreto:** Implementación de su interfaz representando productos concretos.

Pros/Contras del Patrón Abstract Factory

- ➢ + Brinda flexibilidad al aislar a las clases concretas.

- ➢ + Facilita cambiar las familias de productos.

- ➢ - Para agregar nuevos productos se deben modificar tanto las fábricas abstractas como las concretas.