

**TITULO.-**

ABSTRACT FACTORY

**INTEGRANTES.-**

-ATENCIO ARIAS, Jonathan Jeffery

-PARIONA PEREZ, Diego

-SANCHEZ ALIAGA, Becker Luis

-VALENCIA NAVARRO, Jean Pierre

-ZURITA TADEO, Paul

-ZACARIAS, Kevin

**CURSO.-**

METODOLOGIA DE LA PROGRAMACION

**PROFESOR.-**

CORONEL CASTILLO, Eric Gustavo

**AULA Y TURNO.-**

102A – TARDE

**LIMA – PERÚ**

2017- ll

**Abstract Factory** (Fábrica Abstracta)

**INDICE**

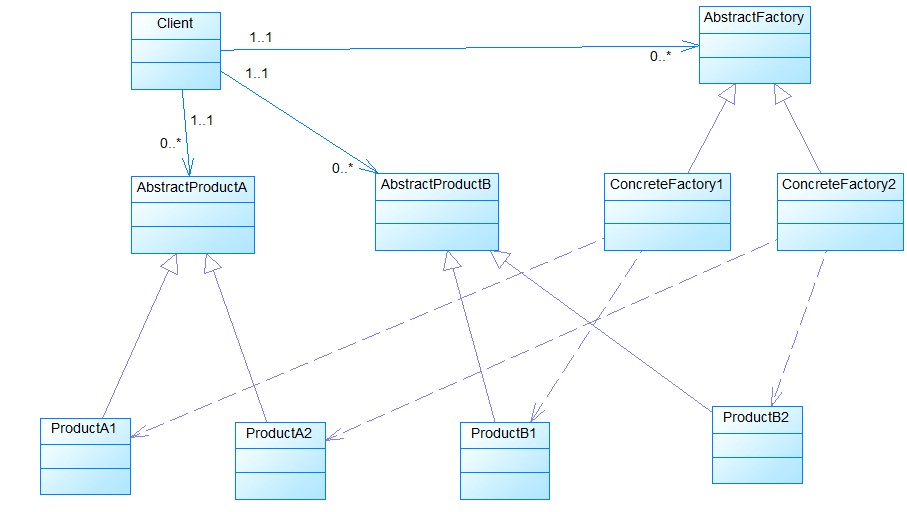
1. Introducción…………………………………………………………………………………………………….2
2. Estructura………………………………………………………..……………………………….…………..…3
3. Características………………………………………………………………….………………………………4
4. Problemática……………………………………………………….…………………………………………..5
5. Resolución………………………..…………………………………………………………….………………6
6. Imágenes del video………………………………………………………………………………………….7
7. Conclusiones………………………………………………………………………………..………………….9
8. Anexos……………………………………………………………………………………………………………11

**INTRODUCCION.-**

Los patrones de diseño son soluciones que se puede aplicar a problemas recurrentes en el diseño de software, los mismos cubren aspectos como la creación e interacción de objetos así como la comunicación entre ellos, además estos nos hacen más fácil el reutilizar componentes de software basándose en técnicas ya probadas una y otra vez en distintas aplicaciones pudiendo realizar diseños de una manera más rápida y simplificada.

En diseño de software, el patrón de diseño **Factory Method** consiste en utilizar una clase constructora (al estilo del Abstract Factory) abstracta con unos cuantos métodos definidos y otro(s) abstracto(s): el dedicado a la construcción de objetos de un subtipo de un tipo determinado.

Es una simplificación del Abstract Factory, en la que la clase abstracta tiene métodos concretos que usan algunos de los abstractos; según usemos una u otra de esta clase abstracta, tendremos uno u otro comportamiento.



**ESTRUCTURA.-**

Las clases principales en este patrón son el *creador* y el *producto*. El creador necesita crear instancias de productos, pero el tipo concreto de producto no debe ser forzado en las subclases del creador, porque las posibles subclases del creador deben poder especificar subclases del producto para utilizar.

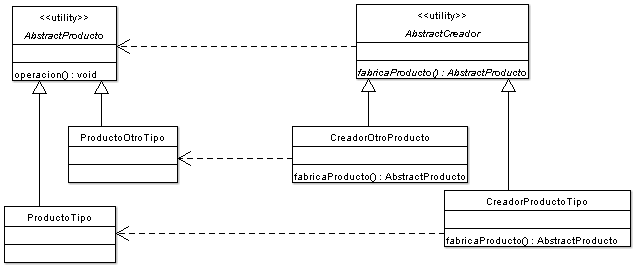
AbstractFactory: declara una interfaz para operaciones que crean un producto abstracto

ConcreteFactory: implementa la operación para crear un producto en concreto

AbstractProduct: declara una interfaz para un tipo de producot en concreto

Product: define un producto que será creado por su ConcreteFactory correspondiente e implementa la interfaz AbstractProduct

Client: usa la interfaz declarada por el AbstractFactory   y AbstractProduct



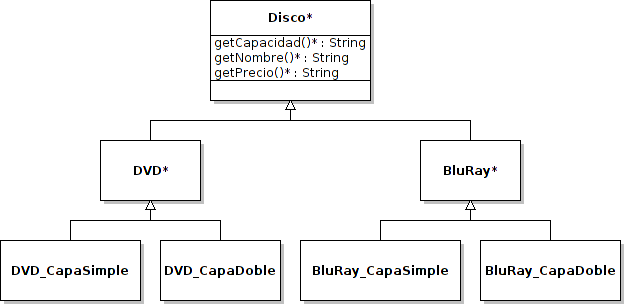
**CARACTERISTICAS.-**

– La creación de un objeto creando su tipo de manera dinámica de forma que no nos veamos atados a una implementación sino a una interfaz haciendo más fácil los cambios futuros.

–  Nos permite jugar con la manera de satisfacer las peticiones hacia una operación determinada, de esta manera es más sencillo cambiar la forma en que se atienden las peticiones desde una operación en concreto.

– Nos permite poder manejarnos entre distintas plataformas de hardware  y software al poder ser capaces de realizar diferentes manejos de las peticiones basándonos en las plataformas.

También es muy usado en casos donde se requiere crear una librería de clases  de productos ya que solo revelamos sus interfaces más no su implementación.



**PROBLEMATICA.-**

Supongamos que disponemos de una cadena de pizzerías. Para crear pizzas disponemos de un método abstracto en la clase Pizzería que será implementada por cada subclase de Pizzería.

Concretamente se creará una clase PizzeríaZona por cada zona, por ejemplo la Pizzería de New York sería PizzeriaNewYork y la de Californía PizzeríaCalifornia que implementarán el método con los ingredientes de sus zonas.

Las pizzas son diferentes según las zonas. No es igual la pizza de New York que la pizza de California. Igualmente, aunque usarán los mismos ingredientes (tomate, mozzarella...) no los obtendrán del mismo lugar, cada zona los comprará donde lo tenga más cerca. Así pues podemos crear un método creador de Pizza que sea

Como vemos utilizamos la factoría abstracta (no las concretas de cada zona, como podría ser IngredientesNewYork o IngredientesCalifornia). Pizza podrá obtener los ingredientes de la factoría independientemente de donde sea. Sería fácil crear nuevas factorías y añadirlas al sistema para crear pizzas con estos nuevos ingredientes. Efectivamente, en este ejemplo *cliente* es Pizza y es independiente de la Factoría usada.

El creador de la Pizza será el encargado de instanciar la factoría concreta, así pues los encargados de instanciar las factorías concretas serán las pizzerías locales. En PizzeríaNewYork podemos tener el método crearPizza() que realice el siguiente trabajo:

**RESOLUCION.-**

**abstract** Pizza crearPizza()

Pizza(FactoriaIngredientes fi);

Pizza crearPizza() {

FactoríaIngredientes fi = **new** IngredientesNewYork();

Pizza pizza = **new** Pizza(fi); *// Uso de la factoría*

pizza.cortar();

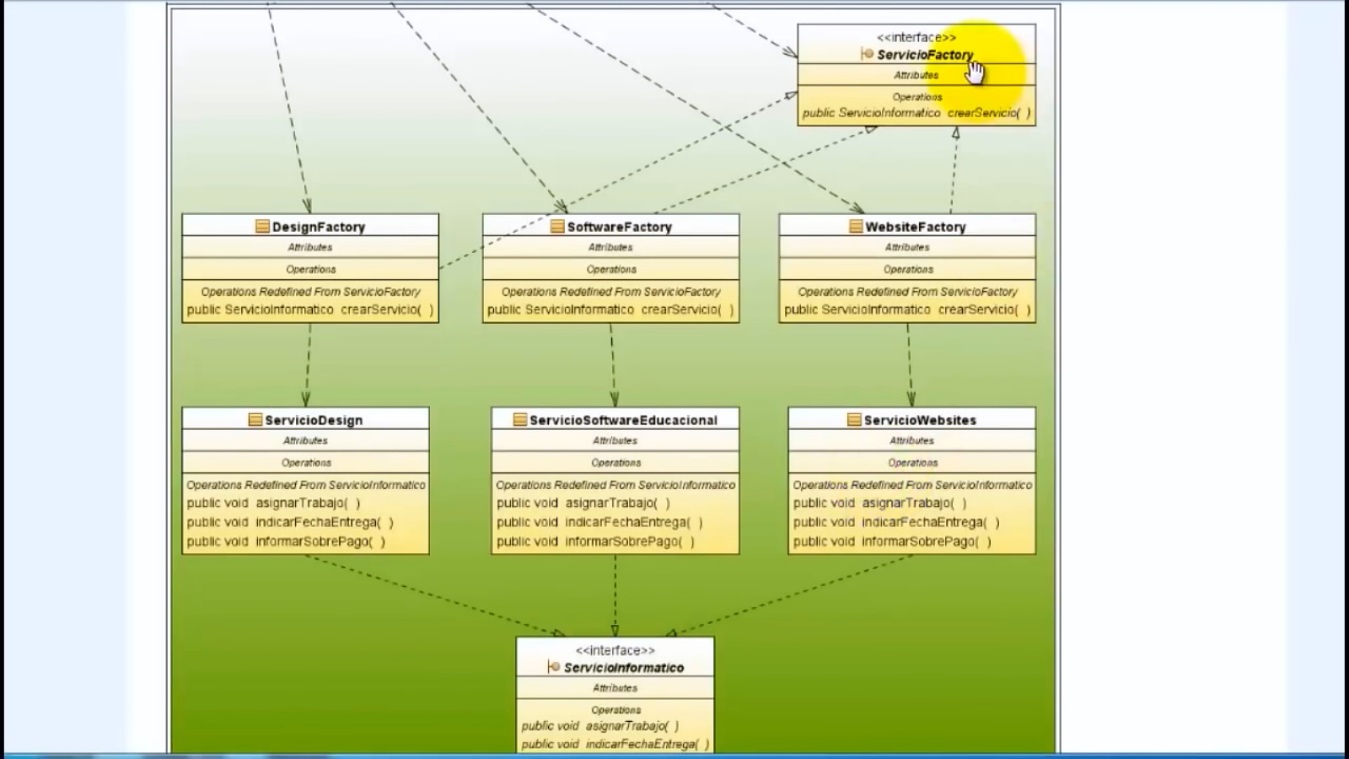
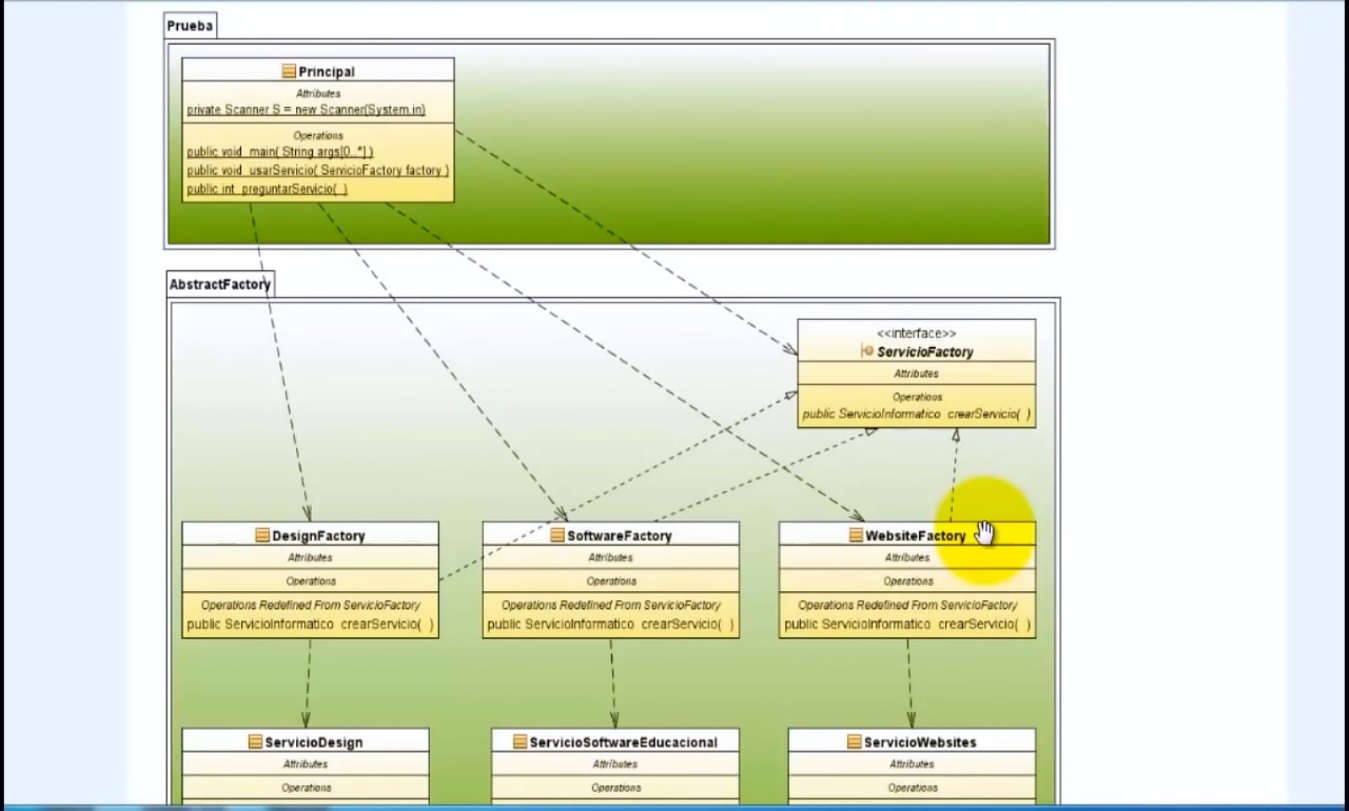
pizza.empaquetar();

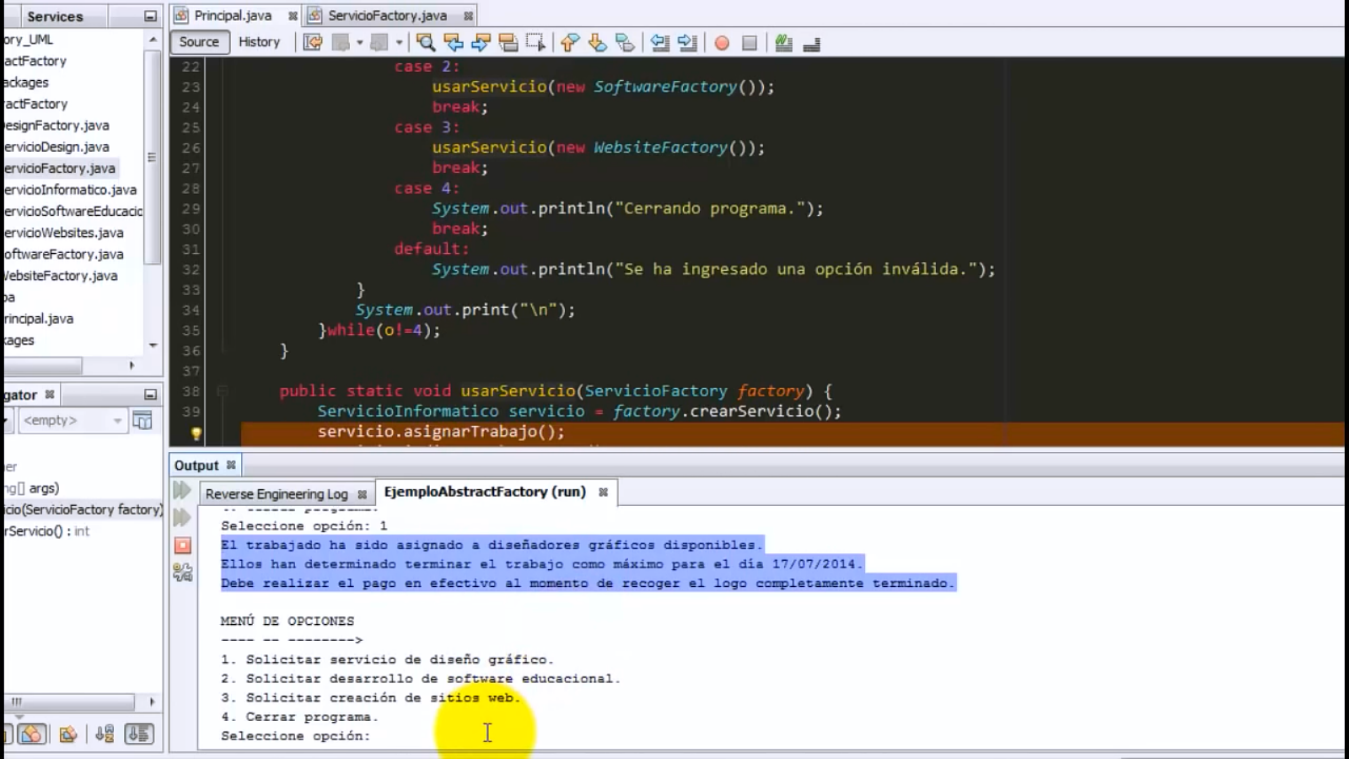
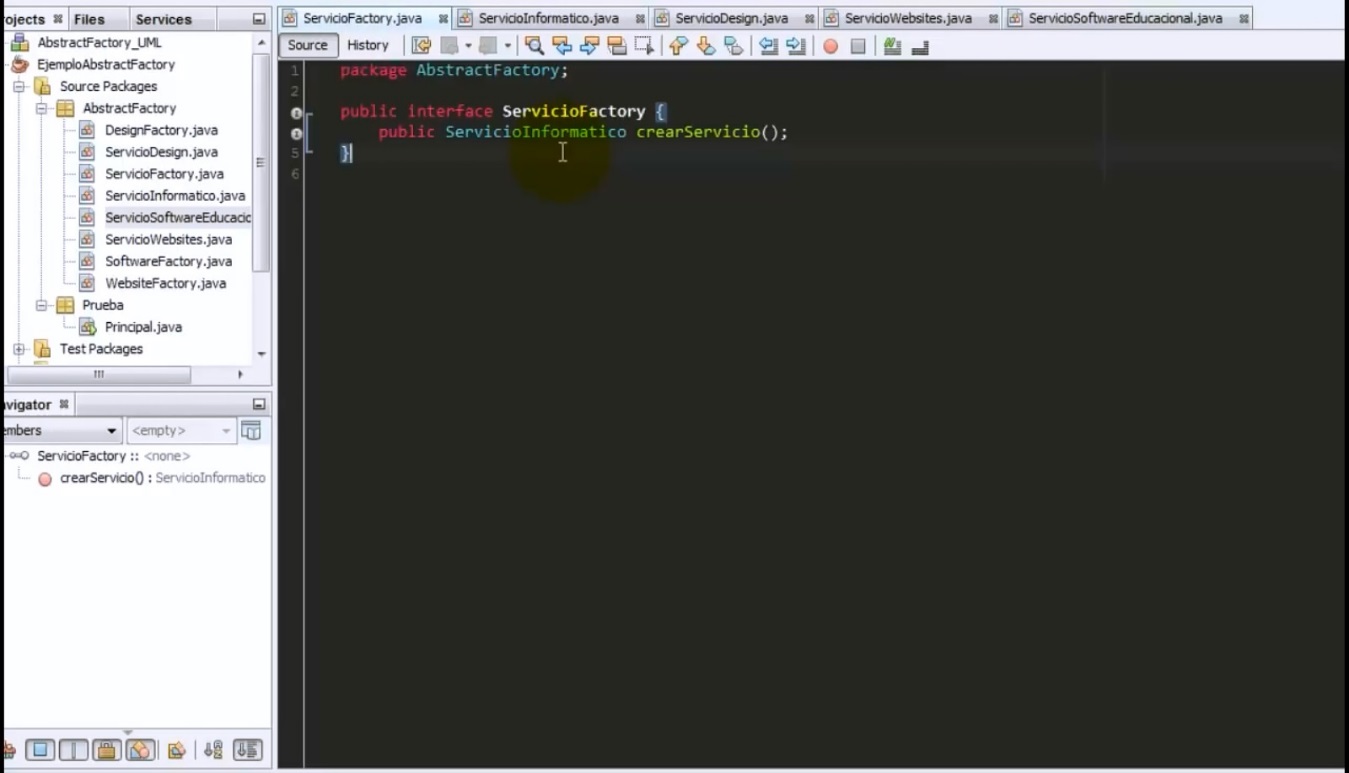
**return** pizza;

}

Como conclusión podemos observar que gracias a la factoría de ingredientes crear una nueva zona, por ejemplo una pizzería en Barcelona, no nos implicaría estar modificando el código existente, solo deberemos extenderlo (uno de los pilares de la Ingeniería del software) ya crearíamos la subclase de Pizzería: PizzeríaBarcelona que al instanciar la factoría solo debería escoger la factoría de Barcelona.

**IMÁGENES DEL VIDEO.-**

****



**CONCLUSIONES.-**

**- Becker.-**

AbstractFactory nos brinda una facilidad de reutilizar componentes de software basándose en técnicas ya probadas una y otra vez en distintas aplicaciones pudiendo realizar diseños de una manera más rápida y simplificada.

**- Diego.-**

El método Factory: es un creador de objetos, además de definirse como interfaz donde las subclases son los creadores de los objetos

Es como tener una fábrica que puede crear diferentes productos donde nosotros podemos mandar a la fábrica a crear un producto, pero los que en realidad se encargan son los elementos internos.

La fábrica toma la decisión dependiendo del objeto que va a crear para elegir a la subclase que va a tomar la responsabilidad de la creación del producto.

En su estructura se tiene el Product: quien define la interfaz de los objetos que crea el método factoría

Concrete product: es el que implementa la interfaz del producto.

Creator: es donde se declara el método factoría que devuelve un objeto de tipo product.Puede definir una implementación por defecto de dicho método, que devuelve un objeto de algún producto concreto .también se puede llamar al método factoría para crear un objeto de tipo product.

**-Kevin.-**

Finalizando la estructura de Abstract Factory va por clases, las cuales serian; el cliente, las factorías concretas, el producto abstracto y producto concreto.  
Estas son las clases principales, intervienen el creador y el producto. El individuo necesita implementar inicios en su creacion, aunque la creacion no de ser forzada por el sujeto concreto y que el creador debe enmarcar bien la idea que quiere transmitir por medio de la clase a utilizar.

**- Zurita.-**

Usando este lenguaje es más complejo pero mejor tipo de diseño.

**- Valencia.-**

Aun no llegamos a dominarlo pero por el simple hecho de que este lenguaje sea el mejor tipo para programas un poco más del tipo de empresas y eso casi como todos los anteriores agrada mucho saber más de este tema de leguajes de programación.

**- Atencio.-**

Según todo lo entendido anteriormente, se llega a concluir que AbstractFactory nos proporciona a nosotros una interfaz para crear objetos relacionados que depende entre sí mismos. Esta clase constructora abstracta con unos métodos definidos, el dedicado a la construcción de objetos de un subtipo de un tipo determinado. Y por último, se puede utilizar este patrón cuando se define una clase a partir de la que se crearán objetos pero sin saber de qué tipo son, siendo otras subclases las encargadas de decidirlo.

**ANEXOS.-**

**-http://shop.oreilly.com/product/9780596007126.do**

-https://www.tutorialspoint.com/design\_pattern/abstract\_factory\_pattern.htm

-https://informaticapc.com/patrones-de-diseno/abstract-factory.php

-https://dzone.com/articles/design-patterns-abstract-factory