**“AÑO DEL BUEN SERVICIO AL CIUDADANO”**



**TEMA : Patrones de diseño Factory (fabrica)**

**DOCENTE : Coronel Castillo Eric Gustavo**

**ESTUDIANTES :**

* **Espinoza Ponciano Denis**
* **Señas Sandoval Valeria**
* **Ventocilla Gomero Fanny**

**CURSO : Programación I**

**CARRERA : Ingeniería de sistemas**

**CICLO : Tercer Ciclo**

**2017**

**LIMA – PERÚ**

**RESUMEN EJECUTIVO**

**IDENTIFICACION DEL PROBLEMA**

* Dificultad al modificar un software, de modo que al cambiar partes del código se manifiestan variedad de errores.
* Molestia e incomodidad al momento de entender los códigos complejos.
* No hay un orden establecido en su arquitectura.

Hoy en día se observa mucho desorden al momento de programar, de modo que nadie pude entender el código ejecutado, y si hay algún problema difícilmente se puede localizar el error el uso de patrones de diseño nos facilita al entendimiento de códigos debido a que tiene una arquitectura establecida.

**PROBLEMA**

¿De qué manera se puede usar y derivar el uso de patrones Factory en la arquitectura de un código y en que nos ayuda?

**PLANTEAMIENTO DE LA SOLUCION**

Aplicando métodos en los que ayude a separar las instancias y podamos corregir posibles errores.

Aplicando una jerarquía de clases en las que ayude al mejor entendimiento de las líneas de código.

Solución una clase derivada toma la decisión sobre qué clase instanciar y cómo instanciarla

Una clase necesita instanciar otra clase derivada de una tercera clase, pero no sabe cuál. Factory Method permite a la clase derivada tomar esta decisión

**COMO SE VA A IMPLEMENTAR LA SOLUCION**

Aplicar Patrón de Diseño Factory

**JUSTIFICACION DE LA INVESTIGACION**

Realizamos este proyecto de investigación porque nos ayudará a conocer el uso de patrones Factory y por ende los programadores puedan entenderlo con más facilidad los códigos realizados.

**OBJETIVO GENERAL**

Analizar el uso de Patrones Factory para poder aplicarlo en los códigos que realicemos.

**OBJETIVOS ESPECÍFICOS**

Identificar en qué casos se podría implementar el uso de Patrones Factory.

Mostrar ejemplos con el uso de Patrones Factory.

Mostrar el uso de Patrones Factory como una alternativa de codificar de manera ordenada.

**MARCO TEÓRICO**

**PATRONES FACTORY**

Los patrones de diseño son soluciones a problemas sencillos del desarrollo de software. Hay patrones de diseño que se enfocan en distintas tareas.

Los patrones creacionales ayudan a generar una forma flexible de crear objetos sin tener que recurrir a usar el constructor original de la clase.

Estos patrones entran en la categoría de patrones de creación,, la cual comparten con otros patrones tales como el Singleton, Builder y Prototype .

Los objetos de fabricación (fábricas) tienen la responsabilidad de crear instancias de objetos de otras clases. Tienen además la responsabilidad y el conocimiento necesario para encapsular la forma en que se crean determinados tipos de objetos en una aplicación.

Existen diferentes patrones de fabricación. En este artículo, trataremos Abstract Factory, Factory Method, y Simple Factory. Los dos primeros están incluidos en el catálogo del GoF (presentado y analizado en la entrega anterior) y serán tratados en mayor profundidad.

**FÁBRICAS**

Para comenzar con nuestro estudio debemos definir claramente qué es una factoría o fábrica. De la misma forma que sucede con muchos conceptos en la Ingeniería del Softwareo sistemas, existen varias definiciones para este término (en la entrega anterior hemos experimentado esta situación para el concepto de patrón); por tanto, es fundamental clarificar este concepto para sentar las bases de nuestro estudio posterior.

El término factory (o fábrica) adolece de ser sobreutilizado e impreciso. Mucha gente se refiere a factory para indicar una implementación de Factory Method o de Abstract Factory (patrones definidos en el libro del GoF). Otros sin embargo, se refieren a un objeto que crea instancias de otros, aunque no sigue las reglas de los patrones mencionados anteriormente.

La falta de consenso sobre el término dificulta la comunicación que, como ya hemos visto anteriormente, es uno de los objetivos de los patrones (poder referirse en forma unívoca a una abstracción de mayor nivel que la clase individual, incrementando así el vocabulario de los ingenieros de software y sistemas). Recordemos entonces una importante frase de la entrega anterior:

**LOS PATRONES:** permiten establecer un vocabulario común de diseño, cambiando el nivel de abstracción a colaboraciones entre clases y permitiendo comunicar experiencia sobre dichos problemas y soluciones. Son también un gran mecanismo de comunicación para transmitir la experiencia de los ingenieros y diseñadores experimentados a los más nóveles, convirtiéndose en unas de las vías para la gestión del conocimiento.

Por tanto, es muy importante establecer un significado claro para este término, a los efectos de construir este vocabulario común.

Llamaremos fábrica, factoría o factory a una clase que implemente 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.

**SIMPLE FACTORY**

Crea objetos sin exponer la lógica de instanciación al cliente. Se refiere al objeto recién creado a través de una interfaz común Diagrama.

El corazón del patrón Simple Factory es la clase 'MobileFactory'. Desde el Cliente cuando creamos una instancia de este 'MobileFactory', esta clase cargará y almacenará en caché todas las clases que implementan la interfaz 'IMobile' usando reflexión u otra lógica. Después de eso podemos llamar al método 'GetMobile (string Name)' en la clase 'MobileFactory' que devolverá el objeto especificado a través del parámetro. El Cliente esperará un objeto que implemente la interfaz 'IMobile'.

**METODO FACTORY**

Podemos utilizar este patrón cuando definamos una clase a partir de la que se crearán objetos pero sin saber de qué tipo son, siendo otras subclases las encargadas de decidirlo.

**ABSTRACT FACTORY**

El patrón de diseño “Factory” se enfoca en la creación de una interface que facilite la creación de objetos que se organizan por diferentes subclases. Esto ocurre con frecuencia cuando se hace uso de la herencia. Una clase abstracta se genera conteniendo los atributos generales, y después se crean clases para objetos específicos. Para evitar llamar constructores específicos se deben crear interfaces que nos ayuden en estas tareas.

**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 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.
* Hace fácil el intercambio de familias de productos Porque en el momento de que requiera lo único que se debe hacer, es cambiar o mejorar una línea de código.
* Fomenta la consistencia entre productos. Encapsula la responsabilidad de creación  
  Porque aísla las clases concretas y ayuda a mantener un control sobre las clases de datos

**DESVENTAJAS**

* Se obliga al cliente a definir subclases de la clase "Creador" sólo para crear un producto concreto y esto puede no ser apropiado siempre.
* Puede ser difícil incorporar nuevos tipos de datos, pero esto se puede tener solución, pasando un parámetro a los métodos de creación de productos, pero con esto sacrificaría seguridad.

**IMPLEMENTACION DE SOLUCION**

Brindar información a todos nuestros compañeros y demás personas que lean este informe puedan saber que son los patrones de diseño de fábrica y cómo podemos implementarlo en nuestros códigos de modo que tengamos una estructura de código más definida e establecida, tal que cualquier programador pude comprender nuestro código.

**CONCLUSIONES**

* Los patrones de diseño son soluciones a problemas de diseño de software que se encuentra una y otra vez en desarrollo de aplicaciones del mundo real.
* Los patrones sobre diseño son reutilizables e interaccionan con objetos.
* Simple Factory devuelve instancias de clase que tienen los mismos métodos. Pueden ser instancias de diferentes subclases derivadas, o de hecho pueden ser clases no relacionadas que comparten la misma interfaz. De cualquier manera, los métodos en esta instancia de clase son iguales y se pueden usar de forma intercambiable.
* Method Factory define una interfaz para crear objetos pero dejan que sean las subclases las que deciden que clase instanciar.
* Abstract Factory proporcionan un interfaz para crear familia de objetos relacionadas o que dependen entre si sin especificar su clase concreta.

**BIBLIOGRAFÍA**

* Recuperado en:https:/msdn.microsoft.com/eses/library/bb972258(d=printer).aspx