Juan José López A00381190

Sara Cardona A003

Daniel Escobar A00

Preguntas Guía basado en las referencias:

Libro Design Pattern Tutorial: Sección 1: Design Pattern Overview.

Libro Libro Guía GOF: Sección Introducción – Págs 1 – 11.

1. ¿Cuál es el propósito principal de los patrones creacionales?

* Proporcionan un conjunto de técnicas y patrones de diseño para la creación de objetos en un sistema, lo que permite a los programadores crear objetos de manera más eficiente, y reducir la complejidad del código. Además, ayudan a mejorar la mantenibilidad, la escalabilidad, flexibilidad y la reutilización del código de un sistema, lo que hace que el proceso de desarrollo de software sea más eficiente y productivo.

2. ¿Cuáles son los patrones creacionales más conocidos?

* **Factory pattern**
* **Singleton pattern**
* **Builder pattern**
* **Prototype pattern**

Ejercicio de Diseño 1:

La empresa TeSirvo, que comercializa servicios públicos requiere un modelo detallado que permita gestionar estos servicios para los clientes. Del cliente es importante tener su nombre, identificación, fecha de registro, código de contrato, dirección de facturación, teléfono de contacto, e-mail, estado (Potencial, Paz y salvo, Con Deuda, Moroso, Retirado o Cartera\_Castigada). Con gestionar se espera que se pueda crear, actualizar, inactivar o consultar los servicios adquiridos por un cliente. Los servicios contratados tienen como característica un código de contrato único por cliente, el tipo de servicio (Agua, Gas o Energía), dirección de instalación, fecha de instalación, fecha de facturación y estado (activo o inactivo). Los servicios contratados de Agua, Gas o Energía tienen además el código del medidor y los valores de contador del mes actual y del anterior, estos valores serán actualizados cada mes por el sistema de facturación.

1. Elabore un diagrama de clases que permita Gestionar los servicios, asuma que se espera que el modelo se defina en una arquitectura MVC- Model – View - Controller.



2. Implemente su modelo en su lenguaje de programación OO favorito.

3. Evalúe si su modelo respeta los principios S.O.L.I.D. Explique cómo cumple con cada principio y si no lo cumple indique el porqué, igualmente argumente si considera que el principio no aplica.

* El modelo cumple con el principio de responsabilidad única ya que cada clase tiene una responsabilidad única. La clase cliente tiene la responsabilidad de asociar servicios, y las clases Gas, Agua y Energía tienen una responsabilidad única que se les puede asignar en el futuro.
* Con el principio de abierto cerrado porque las clases que tenemos no necesitan ser modificadas para realizar otras tareas, sino que tenemos clases que extienden de ellas
* Principio de sustitución de Liskov cumplimos con este principio. Tenemos 2 clases abstractas de las cuales sus clases hijas pueden funcionar usando sus métodos sin que generen problemas en el programa
* Principio de segregación de interfaz, cumplimos con este porque tenemos una interfaz que se especifica en una responsabilidad que es la de calcular el valor del contador.
* Principio de inversión de dependencias, cumplimos con este principio porque nuestros módulos de alto nivel, como lo son las clases Servicio o Cliente, no dependen de los de bajo nivel, si cambiamos algo en las clases Agua, Gas o Energía, no afectará a estas clases.

4. Explique qué se puede hacer para que el modelo soporte los principios que aplican al problema.

Ejercicio de Diseño 2:

La empresa TeSirvo, quiere comercializar otros tipos de servicios públicos los cuales también se quiere gestionar, o sea, que se pueda crear, actualizar, inactivar o consultar los servicios adquiridos por un cliente. Los nuevos tipos de servicio (Telefonía e Internet) tienen como característica el código de contrato único por cliente, dirección de instalación, fecha de instalación, fecha de facturación y estado (activo o inactivo). Para Telefonía e internet se tiene tipo de contrato (plan ilimitado o limitado o prepago). En Telefonía si es limitado o prepago tiene una cantidad de minutos locales y larga distancia incluidos y consumidos. Para Internet cantidad de consumo incluido y consumido en Megas.

5. Evalúe si el modelo inicial puede soportar este nuevo requerimiento. Explique si lo soporta cómo lo hace y si no, qué cambios requiere para soportarlo.

El modelo inicial puede soportar este nuevo requerimiento porque no creamos los servicios como un enum sino que los creamos como clases separadas que heredan de servicio. Al tenerlo así fue muy fácil agregar nuevas clases de los servicios que tienen los mismos atributos y simplemente se agregaron dentro de las clases nuevas los atributos que faltaban.

6. Antes de cambiar su diseño, analice el Patrón Fabrica. ¿Puede servir para mejorar el modelo? Explique su decisión al respecto.

Nuestra solución cumple prácticamente en su totalidad con el Patrón Fábrica, pero para mejorar el modelo, podemos agregar una nueva clase que sea nuestra “fábrica de objetos” en donde se crean los objetos de los servicios para que así la clase servicios no los tenga que crear.

7. Implemente el modelo final y presente cuáles fueron los beneficios de utilizar el patrón recomendado.

Los beneficios de utilizar el patrón recomendado fue que en la clase servicio no se crean los objetos de los servicios volviendo el proceso más eficiente.

Ejercicio de Diseño 2:

La empresa TeSirvo, ha encontrado que es importante ofrecer a sus clientes paquetes de servicios donde el cliente puede adquirir una mezcla de servicios de Agua, Gas y Energía, Telefonía e Internet, pero también se espera tener la capacidad de ofrecer comercialmente combinación de uno o varios servicios. Esto implica que el modelo sea capaz de generar por un registro de paquete varios servicios para un cliente y asociarles alguna característica en común, para que al momento de facturar se identifique que hacen parte de un paquete. Se debe tener en cuenta que cuando se inicie el desarrollo de la facturación es importante que sea identificable la existencia de estos paquetes.

8. Investigue si hay algún patrón creacional que pueda contribuir en una solución para que el diseño actual considere este requerimiento final.

* El patrón de fábrica contribuye con la solución ya que se adapta al enfoque requerido, el cual es la creación de objetos externalizado en una clase aparte la cual decidirá qué producto va a crear, para un más óptimo rendimiento de la máquina al momento de correr el software.
* Aplicado a nuestra problemática vamos a tener que para crear servicios necesitaremos hacer uso de ese patrón, creando una clase fabricaServicios que será en donde se crean los objetos dependiendo del servicio elegido.

9. Elabore los cambios en su diseño e implemente la solución.

