Taller 1 - Librería de notificaciones.

Daniel Muñoz

David Henao

Daniel Lopez

Énfasis en especialización 1 y 2

Jesus Andres Hincapie Londoño

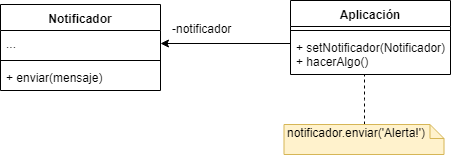
Universidad de Medellín

Medellín Marzo 2021

Enunciado

Suponga que usted trabaja en una empresa de desarrollo y le asignaron la tarea de modificar una librería de notificación que le permite a otros programas notificar a sus usuarios acerca de eventos importantes.

La versión inicial de la librería estaba basada en la clase Notificador, la cual tiene un método enviar. Este método puede aceptar un mensaje como parámetro de un cliente y enviar el mensaje a una lista de correos que fueron pasados al Notificador vía su constructor. Una aplicación de un tercero, que actúa como cliente, es la que debe crear y configurar el objeto Notificador una vez y luego los usa cada vez que algo importante ocurre.



En algún punto, usted se da cuenta que los usuarios de la librería esperan no solo correos de notificación. Muchos de ellos requieren recibir SMS, otros quieren recibir notificaciones vía Facebook y los clientes corporativos quizá quieran recibir notificaciones vía el sistema de mensajería empresarial. Modifique el diseño para esta situación, considerando que la librería se pueda extender para muchas más opciones de notificación de las mencionadas.

Más adelante algún cliente pregunta: ¿por qué no se pueden usar varios tipos de notificación al mismo tiempo? A usted le piden que modifique nuevamente el diseño para considerar dicha situación, tratando de reutilizar lo que se pueda y dejando el diseño flexible para poder incorporar nuevas opciones.

Objetivos

* Realizar un modelo de diseño del enunciado dado que considere buenas prácticas de diseño, utilizando un diagrama de clases de UML.
* Implementar el modelo de diseño planteado, utilizando el lenguaje de programación que soporte orientación por objetos.
* Identificar los patrones de diseño utilizados en el modelo.

Aspectos a tener en cuenta

* El modelo de diseño final debe tener el mayor detalle posible.
* El código debe estar organizado y comentado en los lugares que se requiera explicación.
* No se deben considerar en el modelo los elementos de la interfaz de usuario. Sin embargo, opcionalmente se puede implementar la interfaz gráfica para el ejercicio.
* Se deben entregar un documento con el diseño y la justificación y un enlace al repositorio de código y al video.

Entregables

El documento deberá contener los siguientes elementos:

* Análisis del problema, teniendo en cuenta la justificación de todas las decisiones tomadas en el diseño.
* Diagrama de clases UML con el diseño de la solución (se recomienda presentar el proceso de construcción del diagrama para ir ilustrando las decisiones tomadas)
* Identificación y descripción de los patrones que se ajustan al problema y la solución planteada, justificando adecuadamente la respuesta en términos de los propósitos del patrón y describiendo los roles más relevantes de los elementos del modelo en el patrón.
* Un enlace a un vídeo donde se muestre la funcionalidad de la implementación hecha y se expliquen los beneficios del diseño implementado.
* El enlace al repositorio del código de la implementación.

Solución.

# Análisis.

Al requerirse la extensión de la librería, se toma la decisión de:

1. Crear una interfaz (TipoNotificador), para que se pueda implementar la lógica de enviar los mensajes, de una manera más flexible.
2. Dividir las distintas ( y futuras ) formas de enviar notificaciones en clases, donde se implemente en cada una la forma de cómo se abordará el envío de estas notificaciones.
3. Crear la clase notificador, la cual tendrá una lista de varios notificadores que presentarán distintos comportamientos, desacoplados uno de otros, para que la aplicación pueda usar estos comportamientos libremente.
4. Se tiene una clase FabricaNotificadores, que nos ayuda a crear los tipos de notificadores sin tener que especificar la clase exacta. Esto nos indica que el objeto de tipo notificador puede intercambiarse con flexibilidad y facilidad.

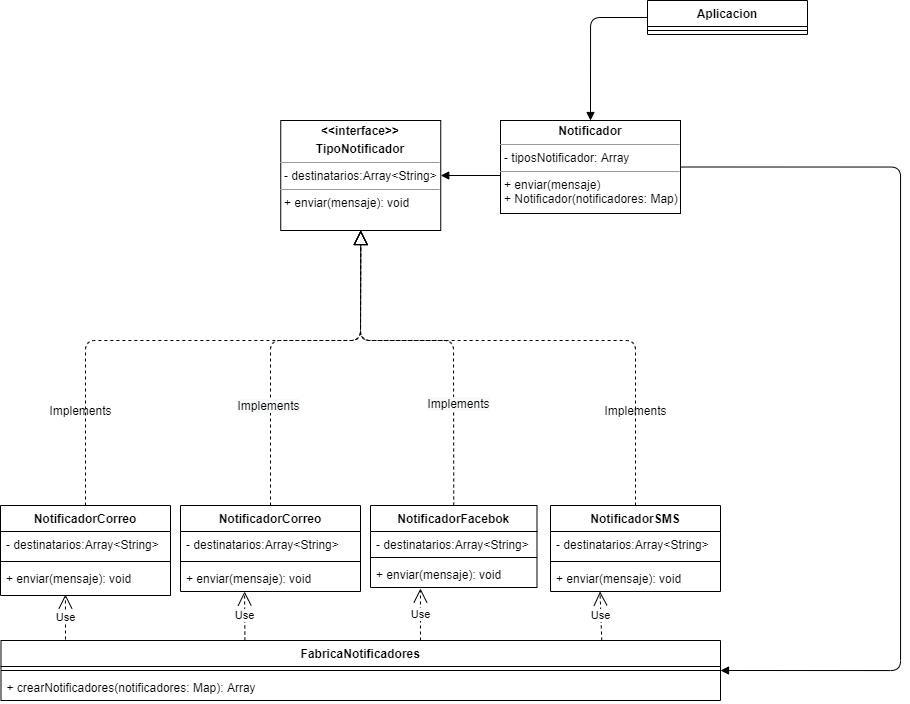
# 

# 

# 

# 

# Diagrama UML.



# Identificación de patrones

En la solución que proponemos, más que todo hacemos énfasis en el principio de segregación de interfaces, donde aumentamos a la cohesión los notificadores existentes o futuros que puedan existir en el sistema, independizando el comportamiento del método enviar en cada notificador.

También se utiliza el patrón de diseño de la factoría, ya que este nos permite la creación de subtipos de notificadores por una clase factoría. La intención de este método es tener una clase a la cual delegar la responsabilidad de creación de los objetos, delegando la responsabilidad a la clase factoría que regrese la clase adecuada para trabajar.