SISTEMA DE NOTIFICACIÓN

Descripción del Problema:
Imagina que estás trabajando en un sistema de notificación para una aplicación de redes sociales. El sistema debe ser capaz de enviar mensajes a los usuarios a través de diferentes canales, como correo electrónico, mensajes de texto (SMS) y mensajes de facebook. Sin embargo, el sistema debe ser lo suficientemente flexible como para permitir a los usuarios elegir qué tipo de notificaciones desean enviar y cómo desean hacerlo. Algunos usuarios prefieren enviar mensajes por correo electrónico, mientras que otros prefieren mensajes de texto o mensajes por facebook.

Actores:
1. Usuario: El usuario de la aplicación de redes sociales.
2. Sistema de Notificación: El sistema que enviará notificaciones a los usuarios.

Objetivos:
1. Permitir a los usuarios elegir el tipo de notificaciones que desean enviar.
2. Permitir a los usuarios elegir cómo desean enviar los mensajes (correo electrónico, SMS, mensajes por facebook).

Escenarios:
1. Los usuarios deben poder seleccionar sus preferencias de notificación a través de la interfaz de usuario de la aplicación.
2. Cuando un usuario realiza una acción que genera una notificación (por ejemplo, recibir un mensaje directo), el sistema de notificación debe enviar la notificación de acuerdo a las preferencias del usuario.

Aplicación de Decorator:
La implementación de la solución al problema es como sigue:
1. Importación de la clase Scanner: El programa comienza importando la clase Scanner para la entrada de datos.
2. Definición de la clase de interfaz *Notificacion*: Se define una interfaz llamada Notificacion con dos métodos abstractos:
    a. mensaje(String mensaje): Para enviar un mensaje.
    b. destinatario(String destinatario): Para especificar un destinatario.
3. Definición de la clase *NotificacionBase*: Esta clase implementa la interfaz Notificacion. En su constructor no hace nada, y los métodos mensaje y destinatario simplemente imprimen mensajes genéricos en la consola.
4. Definición de la clase *NotificacionDecorator*: Esta es una clase abstracta que también implementa la interfaz Notificacion. Tiene un atributo Notificacion notificación y un constructor que acepta una notificación base y la asigna al atributo notificación. Esta clase servirá como base para las notificaciones decoradas.

5. Definición de las clases concretas de decoradores:
    a. *NotificacionFacebook*: Extiende NotificacionDecorator y proporciona una implementación personalizada del método mensaje para enviar mensajes a través de Facebook. También tiene un método destinatario para especificar un destinatario de Facebook.
    b. NotificacionGmail: Extiende NotificacionDecorator y proporciona una implementación personalizada del método mensaje para enviar mensajes a través de Gmail. También tiene un método destinatario para especificar un destinatario de Gmail.
    c. *NotificacionSMS*: Extiende NotificacionDecorator y proporciona una implementación personalizada del método mensaje para enviar mensajes a través de SMS. También tiene un método destinatario para especificar un destinatario de SMS.
6. Definición de la clase *NotificacionTest*: Esta es la clase principal que contiene el método main para ejecutar el programa. Aquí se realiza la interacción con el usuario:
    a. Se crea una instancia de Scanner para recibir entrada del usuario.
    b. Se muestra un menú con las opciones disponibles: enviar notificaciones SMS, Gmail, Facebook o salir del programa.
    c. Se utiliza un bucle do-while para permitir al usuario realizar múltiples acciones hasta que elija salir (opción 4).
    d. Según la opción seleccionada por el usuario, se le pide ingresar un mensaje y un destinatario.
    e. Se crea una instancia de la notificación correspondiente (SMS, Gmail o Facebook) envolviendo la NotificacionBase en el decorador adecuado.
    f. Se llaman a los métodos mensaje y destinatario de la instancia de notificación creada, lo que imprimirá mensajes específicos de cada tipo de notificación.
    g. El bucle continúa hasta que el usuario elige salir (opción 4).

Para la implementación en Java, necesitarás crear las clases mencionadas y permitir que los usuarios elijan sus preferencias a través de la interfaz de usuario. Cuando un evento genere una notificación, el sistema de notificación deberá aplicar los decoradores según las preferencias del usuario y enviar la notificación adecuada. El programa permite al usuario simular el envío de mensajes a través de diferentes canales de notificación y es un ejemplo de cómo el patrón de diseño Decorator se puede utilizar para agregar funcionalidad a las clases existentes de manera flexible y sin modificar su código original.