1. ¿Qué información necesitan las clases EmailSender y SMSSender de la clase Contacto

para realizar su tarea, y qué información recogen? Consideras que incumplen en

principio ISP.

RTA: Las clases EmailSender y SMSSender necesitan acceder a la dirección de correo electrónico y al número de teléfono, respectivamente, del objeto de la clase Contacto para poder enviar el mensaje al destinatario correcto. Sin embargo, la clase Contacto también tiene métodos para obtener y establecer la dirección y el número de teléfono, aunque estas no son necesarias para el envío de mensajes.

Por lo tanto, se puede argumentar que la clase Contacto no cumple con el principio ISP ya que contiene métodos que no son necesarios para todas las clases que la utilizan. En este caso, las clases EmailSender y SMSSender solo necesitan la información de correo electrónico y número de teléfono, respectivamente, pero la clase Contacto también proporciona métodos para obtener y establecer la dirección y el nombre, que no son necesarios para estas clases.

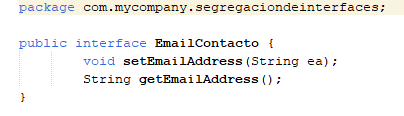
Para cumplir con el principio ISP, se debería dividir la clase Contacto en interfaces más pequeñas y especializadas que proporcionen solo la información necesaria para cada clase que las utilice. Por ejemplo, se podrían crear dos interfaces separadas, una para la información de correo electrónico y otra para la información de teléfono, y luego la clase Contacto podría implementar estas interfaces según sea necesario. De esta manera, las clases EmailSender y SMSSender solo tendrían que depender de las interfaces relevantes y no tendrían que depender de métodos que no necesitan utilizar

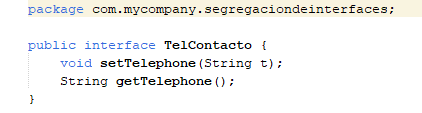
2. Refactoriza las clases anteriores, sustituyendo el parámetro Contacto, por una interfaz.

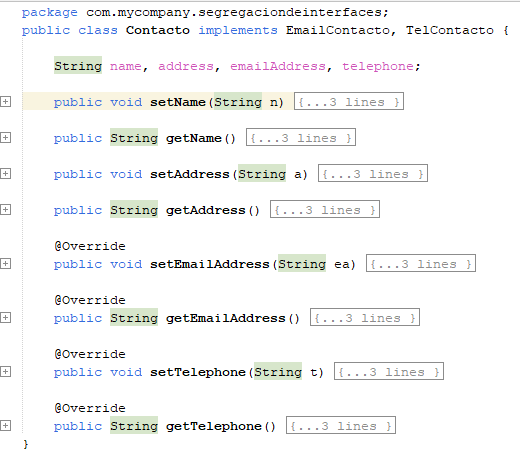
Esta interfaz tendrá los métodos necesarios para acceder a la información que necesita

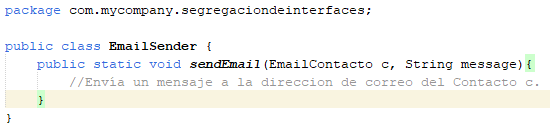
en método. Modifica también la clase Contacto.

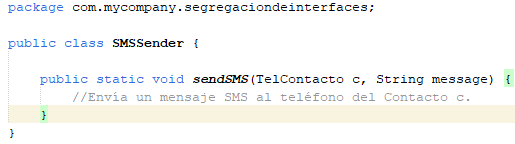
RTA: Para cumplir con el principio ISP, se puede dividir la clase Contacto en dos interfaces separadas, una para la información de correo electrónico y otra para la información de teléfono. Luego, se puede modificar las clases EmailSender y SMSSender para que dependan solo de las interfaces relevantes y no de la clase Contacto completa.











En esta versión, la clase Contacto implementa las interfaces EmailContacto y TelContacto, cada una de las cuales proporciona solo los métodos necesarios para acceder a la información de correo electrónico y teléfono, respectivamente. Las clases EmailSender y SMSSender ahora dependen de estas interfaces en lugar de la clase Contacto completa.

3. Piensa que después de refactorización, la clase GmailAccount (con alguna modificación)

podrá ser enviada a la clase EmailSender pero no a la clase SMSSender.

public class GmailAccount {

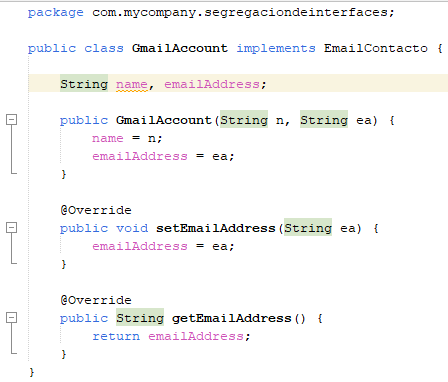
String name, emailAddress;

}

Crea un programa que permita invocar al método sendEmail de la clase EmailSender con

un objeto de la clase GmailAccount.

RTA: Para invocar al método sendEmail de la clase EmailSender con un objeto de la clase GmailAccount, primero necesitamos hacer que la clase GmailAccount implemente la interfaz EmailContacto que creamos anteriormente. Aquí se muestra cómo se podría hacer esto:



Luego, podemos llamar al método sendEmail de la clase EmailSender y pasarle un objeto de la clase GmailAccount de la siguiente manera:

