TALLER PRINCIPIOS SOLID

Punto 5.

Consultas:

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.
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.
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

SOLUCIÓN

1. Las clases EmailSender y SMSSender reciben de la clase contacto un atributo de esta clase, que, en este caso, necesita el correo y el teléfono respectivamente.

   En cuanto al Principio de Segregación de Interfaz (ISP), en principio no se está incumpliendo, ya que `Contacto` proporciona información relevante para ambas tareas, y las clases que las implementan solo utilizan la información que necesitan.

2. Al modificar las clases quedan de esta manera la interfaz y la clase contacto

```
package com.mycompany.taller;

public interface Interfaz_Contacto {

    String getEmailAddress();
    String getTelephone();

}
```

```
package com.mycompany.taller;

/**
 *
 * @author sala8
 */
   public class SMSSender {

   public static void sendSMS(Interfaz_Contacto c,String message){

       String telephone = c.getTelephone();

   }

}
```

```
package com.mycompany.taller;

public class EmailSender {


    public static void sendEmail(Interfaz_Contacto c, String message){

        String emailAddress = c.getEmailAddress();

    }
}
```

```java
package com.mycompany.taller;


public class Contacto implements Interfaz_Contacto {

    String name,address,emailAdress,telephone;


    public void setName(String name) {
        this.name = name;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public void setEmailAdress(String emailAdress) {
        this.emailAdress = emailAdress;
    }

    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }
```

3. Se agrega la clase GmailAccount y en la clase main llamar las funciones

```java
 */
public class GmailAccount implements Interfaz_Contacto {

    String name, emailAddress;


    @Override
    public String getEmailAddress() {
        return emailAddress;
    }

    @Override
    public String getTelephone() {
        return null;
    }
}
```

```java
public class Taller {

    public static void main(String[] args) {

     GmailAccount gmail = new GmailAccount();
     gmail.emailAddress="afbarriosl8@ucatolica.edu.co";
     gmail.name="Felipe";

     EmailSender.sendEmail(c: gmail, message:"asdasd");

    }
}
```