1.1 Protocolo de Cliente

La clase Cliente tiene el siguiente protocolo. ¿Cómo puede mejorarlo?

```
/**
* Retorna el límite de crédito del cliente
*/
protected double lmtCrdt() {...

/**
* Retorna el monto facturado al cliente desde la fecha f1 a la fecha f2
*/
protected double mtFcE(LocalDate f1, LocalDate f2) {...

/**
* Retorna el monto cobrado al cliente desde la fecha f1 a la fecha f2
*/
protected double mtCbE(LocalDate f1, LocalDate f2) {...
```

Los nombres de los métodos explican su propósito, por lo que no se entiende que función cumplen.

Lo mejoraría cambiando los nombres de los métodos:

```java
/**
* Retorna el límite de crédito del cliente
*/

protected double limiteDeCredito() {


}
/**
* Retorna el monto facturado al cliente desde la fecha f1 a la fecha f2
*/

protected double montoFacturadoEntre2Fechas(LocalDate fecha1, LocalDate fecha2) {

}

/**
* Retorna el monto cobrado al cliente desde la fecha f1 a la fecha f2
*/

protected double montoCobradoEntre2Fechas(LocalDate fecha1, LocalDate fecha2) {


}
```

1.2 Participación en proyectos

Al revisar el siguiente diseño inicial (Figura 1), se decidió realizar un cambio para evitar lo que se consideraba un mal olor. El diseño modificado se muestra en la Figura 2. Indique qué tipo de cambio se realizó y si lo considera apropiado. Justifique su respuesta.

Diseño inicial:

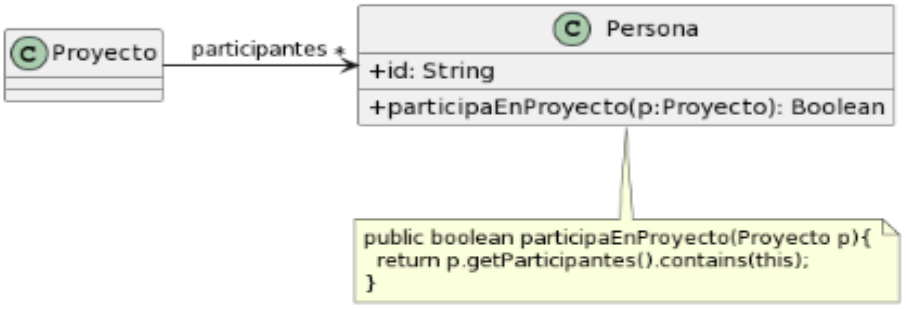

Figura 1: Diagrama de clases del diseño inicial.

Diseño revisado:

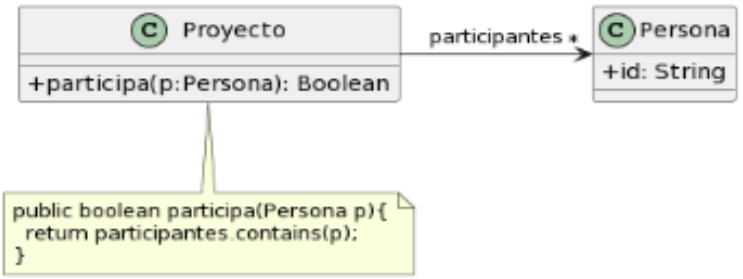

Figura 2: Diagrama de clases modificado.

Los code smells identificados en el primer diseño son:

- Envidia de atributos: La clase Persona tiene envidia de atributos con la clase proyecto, ya que en vez de dejar que la clase proyecto compruebe si la persona participa o no lo calcula por sí misma.
- Clase de datos: La clase Proyecto es una clase que solo contiene datos y no tiene nada de comportamiento.

El segundo diseño me parece correcto, ya que elimina los 2 malos olores mencionados cambiando de lugar el método “participaEnProyecto” a la clase proyecto y renombrarlo “participa”. Lo que hace que la clase Proyecto deje de ser una clase de datos y que la clase Persona sea una clase Dios.

1.3 Cálculos

Analice el código que se muestra a continuación. Indique qué *code smells* encuentra y cómo pueden corregirse.

```
public void imprimirValores() {
    int totalEdades = 0;
    double promedioEdades = 0;
    double totalSalarios = 0;

    for (Empleado empleado : personal) {
            totalEdades = totalEdades + empleado.getEdad();
            totalSalarios = totalSalarios + empleado.getSalario();
    }
    promedioEdades = totalEdades / personal.size();

    String message = String.format("El promedio de las edades es %s y el total de salarios es %s",
promedioEdades, totalSalarios);

    System.out.println(message);

}
```

- Las variables no son muy descriptivas.
- El método es muy largo.

Se pueden corregir dando nombres más descriptivos a las variables y al método y achicando la cantidad de líneas del método haciendo los cálculos independientes a otros métodos.

Corrección:

```
public int calcularTotalEdades() {
int sumaEdades = 0;
 for (Empleado empleado : personal) {
   sumaEdades+= empleado.getEdad();
}
return sumaEdades;
}

public double calcularTotalSalarios() {
double sumaSalarios = 0;
 for (Empleado empleado : personal) {
   sumaSalarios+= empleado.getSalario();
}
```

```
    return sumaSalarios;
    }

    public double calcularPromedioDeEdades() {
     return calcularTotalEdades() / personal.size();.
    }

public void imprimirValores() {
    String message = String.format("El promedio de las edades es %s y el total de salarios es %s",
calcularPromedioDeEdades(), calcularTotalSalarios()) ;

    System.out.println(message);
}
```

Estas correcciones arreglan los malos olores detectados, los métodos y las variables tienen nombres descriptivos y el método que era muy largo y hacía muchas cosas inicialmente, se dividió en métodos más pequeños con una tarea específica.