Ejercicio 1: Algo huele mal.

Indicar qué code smells se presentan en los siguientes ejemplos.

1.1 Protocolo de Cliente.

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

```
/**
* Retorna el límite de crédito del cliente
*/
public 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
*/
private double mtCbE(LocalDate f1, LocalDate f2) {...}
```

Code smells:

- Nombres poco descriptivos.

Solución:

```
/**
* Retorna el límite de crédito del cliente
*/
public double getLimiteDeCredito() {...}

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

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

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:

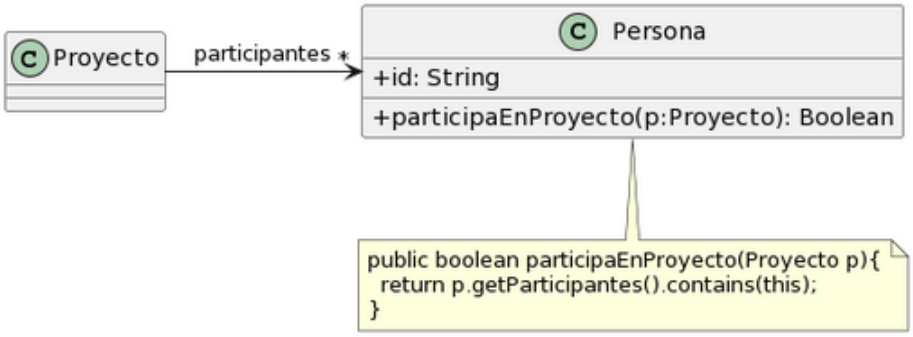

Figura 1: Diagrama de clases del diseño inicial.

Diseño revisado:

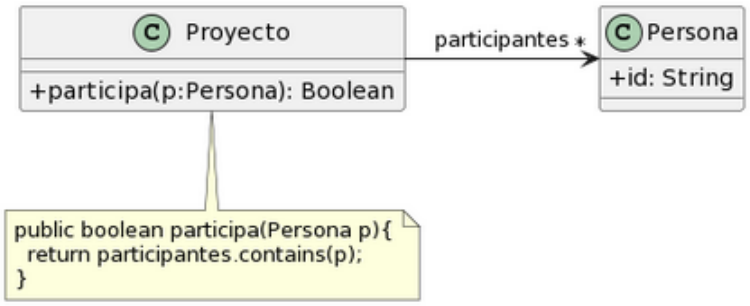

Figura 2: Diagrama de clases modificado.

Code smells:

- Feature Envy.
- Data Class.

En este ejercicio se aplicó el refactoring *Move Method* de manera adecuada, ya que la clase Persona presentaba envidia de atributos en su método participaEnProyecto(p: Proyecto) y la clase Proyecto no tenía comportamiento propio. El objetivo principal del paradigma de programación orientada a objetos es que se puedan encapsular datos junto con los procesos que utilizan los mismos. Por eso lo ideal es que la clase Proyecto consulte a sus datos internos (en este caso los participantes del mismo).

1.3 Cálculos.

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

```java
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);
}
```

Code smells:

- Long Method.
- Nombre de método poco descriptivo.

Es importante corregir el nombre del método, ya que es poco descriptivo respecto a lo que hace. Podría reemplazarse por ejemplo por imprimirEdadesYTotalSalarios().

En segundo lugar, el método imprimirValores() hace demasiadas cosas. Su tarea puede descomponerse en otros métodos más pequeños que estén bien nombrados y hagan uso de herramientas como los streams, haciendo que el código sea más limpio y fácil de entender.

Solución:

```java
public void imprimirPromedioEdadesYTotalSalarios() {
    System.out.println(String.format("El promedio de las edades es %s y el total de salarios es %s", calcularPromedioDeEdades(), calcularTotalSalarios()));
}

public double calcularTotalSalarios() {
    return personal.stream()
        .mapToDouble(empleado -> empleado.getSalario())
        .sum();
}

public int calcularTotalEdades() {
    return personal.stream()
        .mapToInt(empleado -> empleado.getEdad())
        .sum();
}

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