



Bad Smell: Speculative Generality

Integrantes:

Andrea Moruno

Pamela Garces

¿En qué Consiste?

La generalidad especulativa consiste en la creación de código que podría ser utilizado en el futuro, pero que no es utilizado o necesario para la resolución del problema actual. Como resultado, el código no es implementado y se convierte en difícil de entender y de apoyar.

Algunos signos:

Hay una clase, método, campo o parámetro no utilizado.



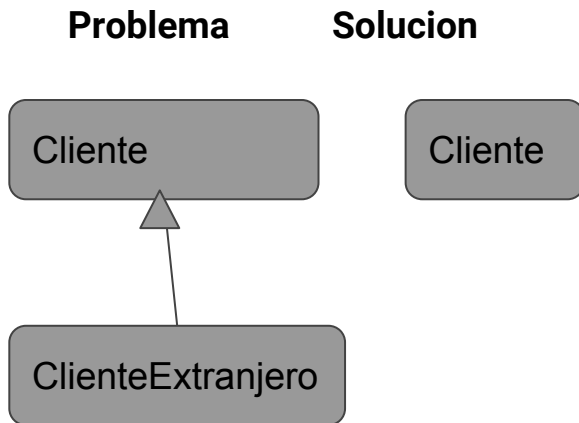
Tratamiento/Soluciones al Bad Smell

- ❖ Para la eliminación de las clases abstractas no utilizadas, se puede usar Colapso de Jerarquía (Hierarchy Collapse).
- ❖ La delegación innecesaria de funcionalidad a otra clase puede ser eliminado a través de Inline Class.
- ❖ Se puede usar Inline Method para eliminar métodos no utilizados.
- ❖ Los métodos con parámetros no utilizados pueden ser observados con Quitar Parámetro (Remove Parameter).
- ❖ Los campos no utilizados pueden ser borrados.



Colapso de Jerarquía (Hierarchy collapse)

- ❖ Cuando se tiene una jerarquía de clases y la subclase es prácticamente igual a la superclase.



- ❖ Porque refactorizar?

El programa tiende a crecer con el tiempo y algunas subclases y superclases son practicamente las mismas. El caso puede darse cuando una subclase tiene un método que es movido a su superclase y en ese momento se tienen 2 clases que se ven prácticamente iguales.

- ❖ Beneficios

La complejidad del programa se reduce. Menos clases significa menos cosas que mantener.

Navegar a traves del codigo es más fácil cuando los métodos están definidos en una sola clase.

Colapso de Jerarquía (Hierarchy collapse)

❖ Como refactorizar

- Seleccionar la clase que sea más fácil de eliminar (superclase o subclase)
- Aplicar las tecnicas “Pull up field” o “Pull up method” si decide eliminar la subclase, o aplique “Push down field” o “Push down method” si decide eliminar la superclase.
- Reemplazar todos los usos de la clase que se ha eliminado con la clase a la cual los campos y métodos se están migrando. A menudo esto sera el codigo para la creacion de clases, variable, parametros y comentarios.
- Eliminar la clase vacia.

❖ Cuando no usar/aplicar

- Si la jerarquía de clases que se está refactorizando, tiene mas de una subclase. Las subclases restantes después de la refactorización deben convertirse en herederos de la clase restante.
- Se debe tener en cuenta la coherencia de la eliminación de clase y que no suceda la violación del principio de sustitución Liskov.

Inline Method

- ❖ Cuando el cuerpo de un método es más obvio que el método en sí, reemplazar las llamadas al método con el contenido del método y eliminar el método.

Problema

```
public class Venta {  
    ...  
    public Double totalVenta() {  
        if (detalleProductos.size() > 0)  
            return ventas();  
        else  
            return 0.0;  
    }  
  
    public Double ventas() {  
        Double total = 0.0;  
  
        for (int i=0;i<detalleProductos.size();i++) {  
            total = total + detalleProductos.get(i).precioVenta -  
descuento;  
        }  
        return total;  
    }  
}
```

Solucion

```
public class Venta {  
    ...  
  
    public Double totalVenta() {  
        Double total = 0.0;  
  
        for (int i=0;i<detalleProductos.size();i++) {  
            total = total + detalleProductos.get(i).precioVenta -  
descuento;  
        }  
        return total;  
    }  
}
```

Inline Method

❖ Porque refactorizar?

Cuando un método simplemente delega en otro método. El uso de delegación no es ningún problema, pero cuando hay muchos de estos métodos, el código se enreda y se complica.

❖ Como refactorizar

- Asegúrese de que el método no está redefinido en las subclases (que no es polimórfico). Si lo es, no es ideal usar esta técnica.
- Buscar todas las llamadas al método. Reemplazar estas llamadas con el contenido del método.
- Eliminar el método.

❖ Beneficios

- El código es más sencillo con la minimización del número de métodos que no son necesarios.

Remove Parameter

Problema

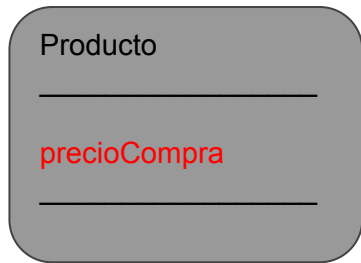


Diagram illustrating the problem: A class (represented by a rounded rectangle) has three parameters listed vertically, separated by horizontal lines. The first parameter is "Producto". The second parameter is "precioCompra" (highlighted in red). The third parameter is an empty line.

Solucion

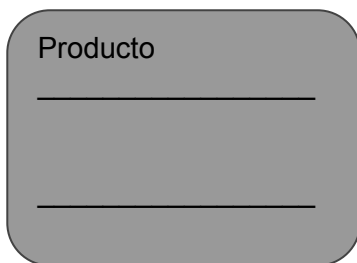


Diagram illustrating the solution: A class (represented by a rounded rectangle) has two parameters listed vertically, separated by horizontal lines. The first parameter is "Producto". The second parameter is an empty line.

❖ Porque refactorizar?

Cada parámetro en una llamada al método obliga al programador a leerlo para averiguar qué tipo de información se encuentra en ese parámetro. Y si un parámetro no se utiliza en el cuerpo del método, no sirve para nada.

❖ Beneficios

Un método contiene solo los parametros que verdaderamente requiere



Remove Parameter

❖ Como refactorizar

- Ver si el método se define en una superclase o subclase. Si es así. Es el parámetro que se utiliza ahí?. Si el parámetro se utiliza en una de las implementaciones, no aplicar esta técnica de refactorización.
- El siguiente paso es importante para mantener el programa funcional durante el proceso de refactorización. Crear un nuevo método, copiando el antiguo y eliminar el parámetro relevante de ella. Reemplazar el código de del método viejo, con la nueva llamada.
- Encuentre todas las referencias al viejo método y reemplazarlas con las referencias del nuevo método.
- Borre el viejo método, No ejecute este paso si el viejo método es parte de una interface pública, en ese caso marque el viejo método con la anotación deprecated.

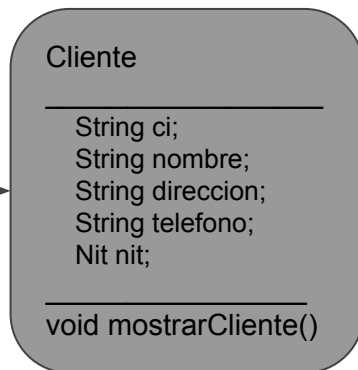
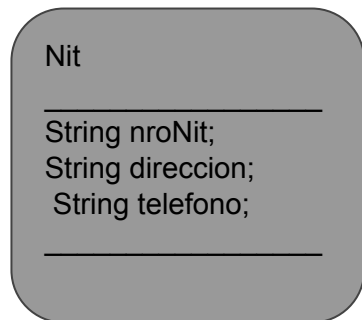
❖ Cuando no usar/aplicar

- Si el método es implementado de diferentes formas en subclases o superclases y su parámetro es usado en aquellas implementaciones, dejar el parámetro tal cual está.

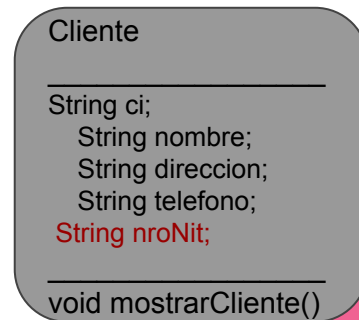
Inline Class

- ❖ Cuando una clase no hace casi nada y no es responsable de nada, y no existen responsabilidades adicionales previstos para dicha clase.
- ❖ La solución es mover todas los atributos y métodos de esa clase a otra.

Problema



Solucion



Inline Class

❖ Porque refactorizar?

Se usa cuando se transfieren los atributos/métodos de una clase a otras, así dejando a la clase con poco que hacer.

❖ Como refactorizar

- En la clase destinatario, crear los campos y métodos públicos presentes en la clase donador. Los métodos deben referirse a los métodos equivalentes de la clase donador.
- Reemplazar todas las referencias a la clase donador con referencias a los campos y métodos de la clase destinatario.
- Probar el programa y asegurarse de que no hay errores. Si no los hay, utilizar el método mover (Move Method) y mover campo (Move Field) para transferir por completo toda la funcionalidad de la clase original a la clase destinatario.
- Continuar hasta que la clase original este vacío. Eliminar la clase original.

❖ Beneficios

- Eliminar clases innecesarias libera memoria y deja código más simplificado.