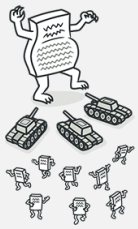
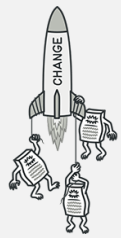
Bloaters



Dispensables

Change Preventers



Couplers

OO Abusers

Long Method

Un método que contiene demasiadas líneas de código.
En general, cualquier método de más de diez líneas debería hacerte dudar de él.

Cómo detectarlo	Refactoring
• Método largo	• Extraer método • Extraer método en otro objeto • Descomponer los condicionales

Long Parameter List

Un método que contiene demasiados parámetros.
En general, cualquier método con más de tres o cuatro parámetros.

Cómo detectarlo	Refactoring
• Más de tres o cuatro parámetros para un método	• Reemplazar por una llamada • Introducir un objeto de parámetro • Preservar objeto original

Data Clumps

Diferentes partes del código contienen grupos idénticos de variables (como parámetros para conectarse a una base de datos).

Cómo detectarlo	Refactoring
• Deja-vu en lista de parámetros	• Introducir un objeto de parámetro • Extraer a clase • Preservar objeto original

Primitive Obsession

Utilizar demasiados tipos primitivos en la implementación. Esto genera un problema de ofuscación del objetivo real de la implementación.

Cómo detectarlo	Refactoring
• Mucho detalle en la implementación • Demasiados campos con tipos primitivos	• Reemplazar con objetos • Reemplazar con clases • Reemplazar con State/Strategy • Extraer a clase

Duplicate Code

Se presenta con varias formas. Por ejemplo la repetición de código en diferentes sitios, o también como métodos con código diferente pero con la misma funcionalidad.

Cómo detectarlo	Refactoring
• Deja-vu en código	• Extraer método • Extraer a clase • Ascender método • Extraer método a template

Lazzy Class

A diferencia de Data Class, se trata de una clase que tiene muy poca responsabilidad y que casi no se utiliza.

Cómo detectarlo	Refactoring
• Clases muy pequeñas no usadas mucho • Clases que solo delegan a otras clases	• Clase inline • Colapsar la jerarquía

Comments

Usar comentarios para clarificar partes del código evidencia que el código está mal implementado. Generalmente se pueden eliminar con renombrados.

Cómo detectarlo	Refactoring
• Comentarios explicando variables, métodos o pasos dentro de un método	• Extraer método • Renombrar método o campo • Introducir una Aserción

Data Class

Es una clase que no es un POJO y solo contiene propiedades y getters y setters para acceder a la propiedades. No tiene lógica.

Cómo detectarlo	Refactoring
• Clase con propiedades solamente	• Mover método • Encapsular un campo • Encapsular una colección

Magic Number

Son valores (numéricos, literales) que aparecen en el código (asignaciones o comparaciones) y que, sin un contexto previo, no se sabe el significado de ese valor.

Cómo detectarlo	Refactoring
• Valores literales en el código, a menudo duplicados en varios lugares	• Reemplazar el Magic Number con una Constante

Shotgun Surgery

Se da cuando para hacer una pequeña modificación, se requiere hacer muchas pequeñas modificaciones en muchas clases.

Cómo detectarlo	Refactoring
• Cuando se modifican muchas clases para un cambio aparentemente simple	• Mover método • Mover campo • Clase inline

Divergent Change

Se da cuando haces demasiados cambios continuados en una misma clase. Esa clase tiene demasiada responsabilidad y por eso le afectan muchos cambios.

Cómo detectarlo	Refactoring
• Deja-vu de la clase que estás modificando	• Extraer clase

Uncommunicative Name

El nombre no describe correctamente la intención del método o campo. Los nombres vagos hacen más difícil de entender el código y pueden llegarse a malinterpretar.

Cómo detectarlo	Refactoring
• Nombres de una letra • Abreviaciones técnicas de nombres	• Renombrar el método, clase, campo o variable

Feature Envy

Un método accede demasiado al código de otro objeto para implementar funciones que no le competen.

Cómo detectarlo	Refactoring
• Muchas llamadas a objeto.getXXX	• Extraer método • Mover método

Inappropiate Intimacy

Una clase utiliza atributos y métodos internos de otra. Suelen depender entre ellas.

Cómo detectarlo	Refactoring
• Clases que se suelen utilizar combinadas	• Cambiar la asociación bidireccional a unidireccional • Mover método • Mover atributo • Ocultar la delegación

Message Chains

Invocaciones de métodos encadenadas entre sí una detrás de otra, como **obj.doA().doB().doC().doD()**.

Cómo detectarlo	Refactoring
• Expresión con dos o más puntos	• Extraer método • Mover método • Ocultar la delegación

Alternative Classes with Different Interfaces

Dos clases hacen lo mismo pero implementando métodos con diferentes nombres.

Cómo detectarlo	Refactoring
• Nombres de clase con la misma intención	• Extraer método • Renombrar método • Mover método

Temporary Fields

Se da cuando una clase tiene campos temporales que solo se rellenan bajo ciertas circunstancias, el resto de veces están vacíos.

Cómo detectarlo	Refactoring
• ¿Qué es ese campo de ahí? • Comprobar que los campos no estén siempre vacíos	• Extraer clase • Introducir objeto Null

Switch Statements

Un método que contiene demasiadas secuencias if else o una sentencia switch muy compleja. Dificultan la legibilidad del código.

Cómo detectarlo	Refactoring
• Sentencias switch • Estructuras if-elseif-elseif	• Reemplazar condición con métodos explícitos • Reemplazar con State/Strategy • Reemplazar con polimorfismo • Introducir objeto Null