¿Qué es un "Code Smell"?

Introducción a la calidad del código en Java

Los "Code Smells" son indicadores en el código fuente que sugieren posibles problemas de diseño o implementación. No son necesariamente errores, pero señalan áreas donde el código podría mejorarse para mayor calidad y mantenibilidad.

- Detecta problemas
- Mejora la calidad
- Favorece las buenas prácticas

Code Smells: Definición y Características

Definición

Un **"Code Smell"** es un indicio en el código fuente que sugiere que podría existir un problema más profundo en el diseño o implementación del software.

No son errores técnicos, sino violaciones de principios fundamentales de diseño que pueden dificultar el mantenimiento del código a largo plazo.

Importancia

- Facilita el mantenimiento y evolución del código
- Reduce la deuda técnica y costos a largo plazo
- Mejora la legibilidad y comprensión del código
- Previene errores futuros y facilita las pruebas

Características

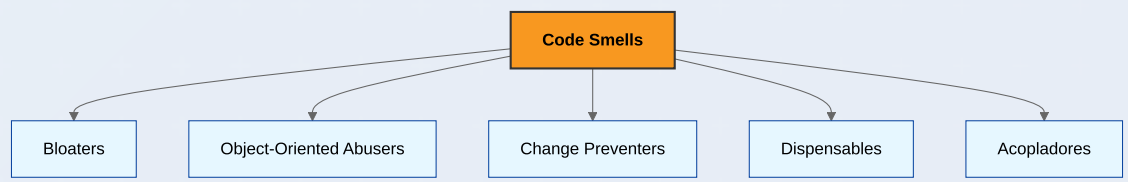
Síntoma, no enfermedad
Indican problemas potenciales, no necesariamente errores

Detectables
Se pueden identificar mediante revisiones de código o herramientas de análisis

Refactorizables
Pueden corregirse mediante técnicas de refactorización específicas

Subjetivos
Su importancia puede variar según el contexto y el proyecto

Categorías Principales



"Un code smell es una superficie característica que puede indicar un problema más profundo." — Martin Fowler

</> Ejemplos de Code Smells en Java

Método Largo

Problema:

```
public void procesarDatos(List<Cliente> clientes) {
    // Más de 100 líneas de código
    for(Cliente c : clientes) {
        // Validaciones
        if(c.getNombre() == null || c.getNombre().isEmpty()) {
            System.out.println("Error: nombre vacío");
            continue;
```

Solución:

```
public void procesarDatos(List<Cliente> clientes) {
    for(Cliente c : clientes) {
        if(esClienteValido(c)) {
            aplicarDescuento(c);
            // Otras operaciones llamando a métodos específicos
        }
    }
```

Duplicación de Código

Problema:

```
public class GestorUsuarios {
    public void registrarUsuario(String nombre, String email) {
        // Validación de email
        if(!email.contains("@") || !email.contains(".")) {
            throw new IllegalArgumentException("Email no válido'
        }
        // Lógica para registrar usuario
```

Solución:

```
public class GestorUsuarios {
    public void registrarUsuario(String nombre, String email) {
        validarEmail(email);
        // Lógica para registrar usuario
        System.out.println("Usuario registrado: " + nombre);
    }
```

Nombres Poco Descriptivos

Problema:

```
public class Data {
    private List<Object> data;

    public void process() {
        for(Object o : data) {
            if(o instanceof String) {
                String s = (String)o;
```

Solución:

```
public class ProductoInventario {
    private List<Object> inventarioItems;

    public void procesarInventario() {
        for(Object item : inventarioItems) {
            if(item instanceof String) {
                String codigoProducto = (String)item;
```

Conclusión

- **Identificar** code smells es el primer paso para mejorar la calidad del código
- **Refactorizar** regularmente ayuda a prevenir la acumulación de deuda técnica
- **Revisar código** en equipo facilita la detección de problemas
- **Invertir** en aprender patrones de diseño ayuda a evitar code smells