Ejemplos reales de code smells

Versión 1.1
Mayo 2020

1 Introducción

El siguiente es un listado de ejemplos de code smells extraídos directamente de código de sistemas reales, junto a algunas preguntas conceptuales autoevaluativas. Observá cada ejemplos prestando atención a los siguientes aspectos:

1. No es necesario conocer el lenguaje o la tecnología en detalle para detectarlo
2. De igual forma, no es necesario conocer el dominio para entenderlo
3. Es posible proponer un refactor "a ciegas", pero para verdaderamente saber si vale la pena necesitaremos la información mencionada en los puntos anteriores.

2 Los ejemplos

2.1 Type Test (Java)

```java
/**
 * chequea que el usuario tenga el permiso para modificar el tipo
 * de movimiento indicado
*/
void authModificacionPorTM(Movimiento aj) {
    if (COBRANZA_APERTURA.esTipoDe(aj)) {
        authz(MODIFICACION_APERTURA_COBRANZA);
    } else if (COBRANZA_IMPUTACION.esTipoDe(aj)) {
        authz(MODIFICACION_CAMBIO_IMPUT_COBRANZA);
    } else if (COBRANZA_DIRECTO.esTipoDe(aj)) {
        authz(MODIFICACION_AJUSTE_DIRECTO_COBRANZA);
    } else if (DEPOSITO_APERTURA.esTipoDe(aj)) {
        authz(MODIFICACION_APERTURA_DEPOSITO);
    } else if (DEPOSITO_IMPUTACION.esTipoDe(aj)) {
        authz(MODIFICACION_CAMBIO_IMPUT_DEPOSITO);
    } else if (DEPOSITO_DIRECTO.esTipoDe(aj)) {
        authz(MODIFICACION_AJUSTE_DIRECTO_DEPOSITO);
    }
}
```

Para pensar: ¿qué pensás de los nombres escogidos en el código? ¿Son un code smells en sí mismo? ¿Qué abstracción está faltando? ¿Qué refactor encararías? ¿Cómo distribuirías las responsabilidades?

2.2 Long Parameter List (C#)

```
/* este es un envío del mensaje DoGetPartidasPendientes al objeto partidaService. Por otro lado, data es un objeto */
partidaService.DoGetPartidasPendientes(
            data.EmpresasResource,
            data.EmpresaSeleccionada,
            data.TipoDeCobranzaSeleccionado,
            data.FechaDesde, data.FechaHasta, data.ImporteDesde,
            data.ImporteHasta, data.ModoMoneda, modoEntidades,
            data.ModoPartidas, data.ModoUsuarios,
            todasEntidades, todasLasMonedas,
            todasLasPartidas, usuarios,
            data.BuscarSinClasificar);
```

Para pensar: ¿cuáles problemas identificás? ¿Son necesarios 16 parámetros? ¿Por qué? ¿Qué abstracción o abstracciones están faltando? ¿Hay abstracciones que no estemos utilizando adecuadamente?

2.3 Primitive obsession

(Java)

```
public interface Criterio {
	int implementarCriterio(int totalPuntos, int totalCorrectos, int n );
}
```

(Scala)

```
abstract class Compra {
 var datosDeEntradas:List[(Char,Int,Int,Int)] =  ...
}
```

Para pensar: ¿Qué problemas traería el diseño presentado? ¿Con qué cualidades de diseño están asociados? ¿Cómo se podrían evitar?

2.4 Duplicated Code (C#)

```
if (entidadModo.Equals(ModoOpciones.LISTA)) {
   String codEntidades = "";
   bool firstTime = true;
   foreach (var entidad in entidades) {
       if (firstTime) {
           firstTime = false;
       } else {
           codEntidades += ",";
       }
       codEntidades += entidad.codigo;
   }
   parametros.Add("codEntidades", codEntidades);
}
```

```
if (usuarioModo.Equals(ModoOpciones.LISTA)) {
   String nombresUsuarios = "";
   bool firstTime = true;
   foreach (var user in usuarios) {
       if (firstTime) {
           firstTime = false;
       } else {
           nombresUsuarios += ",";
       }
       nombresUsuarios += user.username;
   }
   parametros.Add("usuarios", nombresUsuarios);
}
```

```
if (monedaModo.Equals(ModoOpciones.LISTA)) {
   String codMonedas = "";
   bool firstTime = true;
   foreach (var moneda in monedas) {
       if (firstTime) {
           firstTime = false;
       } else {
           codMonedas += ",";
       }
       codMonedas += moneda.codigo;
   }
   parametros.Add("codMonedas", codMonedas);
}
```

Para pensar: ¿Qué algoritmo está oculto y repetido a través de los ejemplos de código? ¿Podrias reescribirlo de forma genérica y usarlo? ¿Cómo?

2.5 Long method (Java/Android)

```
@Override protected void onCreate(Bundle savedInstanceState) {
    Logger.debug(this, "onCreate");
     super.onCreate(savedInstanceState);

     editType = EditType.valueOf(getIntent().getStringExtra(IntentHelper.EDIT_TYPE));
     int positionInList = getIntent().getIntExtra(IntentHelper.CONTACT_ID_IN_LIST, -1);
     String previousValue = getIntent().getStringExtra(IntentHelper.EDIT_PREVIOUS_VALUE);

     getSupportActionBar().setDisplayHomeAsUpEnabled(true);
     getSupportActionBar().setDisplayShowHomeEnabled(false);
     getSupportActionBar().setHomeButtonEnabled(true);
     getSupportActionBar().setTitle(editType.getEditTitle());
     findViews();


     isPah = positionInList == IntentHelper.PAH_POSITION;
     Logger.debug(this, "Current position in list for edit " + positionInList);
     Logger.debug(this, "Current Size of the list ist: " +
           taggApp()
            .getUserModel()
            .getFullNotificationSettings()
            .getContactsNotifInfo().size());
     if (isPah) {
         cnd = taggApp().getUserModel().getFullNotificationSettings().getUserNotifInfo();
     } else {
         cnd = ListUtils.getAtOrLast(positionInList, taggApp().getUserModel().getFullNotificationSettings().getContactsNotifInfo());
     }


     text = (ClearableEditText) findViewById(R.id.clearable_edit_text);

     restorePreviousValue(previousValue, text);
     text.setOnChangeListener(watcher);
     enabledHook();


     text.addTextChangedListener(watcher);

     text.setHint(editType.getHint());
     Logger.debug(this, "onCreate end");
}
```

Para pensar: Notá que hay algunas líneas en blanco, ¿por qué quien desarrolló este código las insertó? ¿Qué nos quiso decir? ¿Qué problema tiene la estrategia que planteó? ¿Cuántos objetos o métodos podrías extraer? ¿Cómo? ¿Por qué es un método largo? ¿Es estrictamente por la cantidad de líneas? ¿Hay una cantidad de líneas que taxativamente nos permita identificar a este code smell? ¿Por qué?

Por otro lado, más allá del *long method*, ¿qué otro u otros *code smells* presenta?

Finalmente: ¿por qué pensas que el método se llama onCreate? ¿Habrá sido una decisión de quienes diseñaron el sistema o estará dada por algún otro factor? ¿Podrías relacionarlo con los concepto de Framework y biblioteca?

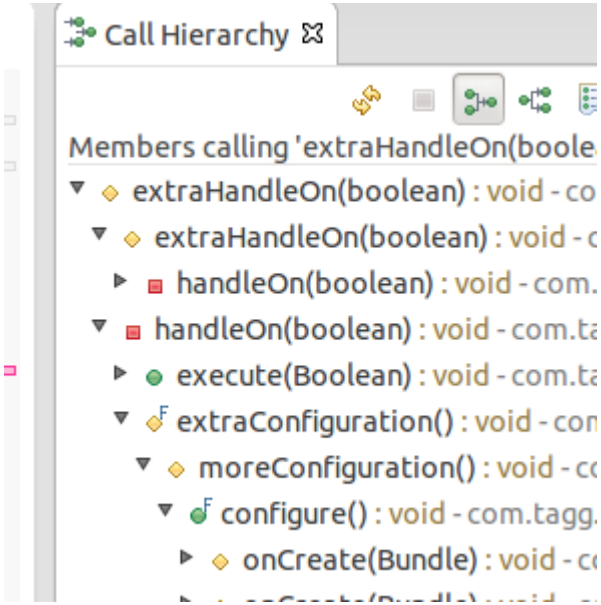
2.6 Feature Envy (Java)

```
class Negocio {
  double calcularPrecio (Prenda prenda) {
     return (this.valorFijoDelNegocio + prenda.getPrecioBase(
               )) * prenda.getTasaImportacion() * prenda.getCoeficienteMarca();
  }
 //...
}
```

Para pensar: ¿Cuál es esa responsabilidad de la que el `Negocio` está intentando tomar control? ¿Por qué? ¿Dónde debería estar? Intentá listar al menos tres criterios diferentes decidir cuándo asignarle una responsabilidad a un objeto.

3 Un code smell sin nombre

¡Y para que huela mal no tiene que tener un nombre!



Para pensar: ¿Por qué este código *huele mal*? Si tuvieras que "inventar" un code smell para describir el problema, ¿cómo lo llamarías? ¿Cuál sería la solución?