**Nombre:** Génesis Juliana Riera Naranjo

**Paralelo:** 2

* **Feature Envy – Característica de la envidia**

Este tipo de mal olor se da cuando cuando un método está accediendo a los datos de otro objeto sin tomar en cuenta sus propios datos.

**Técnicas de Refactorización**

Move Method

Extract Method

**Ejemplo:**

**Con mal olor**

Public class Telefono {

Cadena final privada sin formato Número;

Teléfono público (String unformattedNumber ) {

this .unformattedNumber = unformattedNumber ;

}

public String getAreaCode () {

return unformattedNumber.substring ( 0 , 3 );

}

public String getPrefix () {

return unformattedNumber.substring ( 3 , 6 );

}

public String getNumber () {

return unformattedNumber.substring ( 6 , 10 );

}

}

Cliente de clase pública ...

Teléfono privado Teléfono móvil;

public String getMobilePhoneNumber () {

return "(" +

**mobilePhone.getAreaCode ()**  + ")" +

**mobilePhone.getPrefix ()**  + "-" +

**mobilePhone.getNumber ()** ;

}

**Sin mal olor**

Public class Telefono {

Cadena final privada sin formato Número;

Teléfono público (String unformattedNumber ) {

this .unformattedNumber = unformattedNumber ;

}

private String getAreaCode () {

return unformattedNumber.substring ( 0 , 3 );

}

private String getPrefix () {

return unformattedNumber.substring ( 3 , 6 );

}

private String getNumber () {

return unformattedNumber.substring ( 6 , 10 );

}

public String toFormattedString () {

**return "(" + getAreaCode () + ")" + getPrefix () + "-" + getNumber ();**

}

}

Cliente de clase pública ...

Teléfono privado Teléfono móvil;

public String getMobilePhoneNumber () {

return **mobilePhone.toFormattedString ();**

}

* **Primitive Obsession – Obsession Primitiva**

Este tipo de mal olor existe cuando en un código se están usando tipos de datos primitivas en vez de hacer uso de objetos, por lo general este mal olor se encuentro cuando se realizan tareas simples.

**Técnicas de Refactorización**

Replace type code with class

Replace type code with subclases

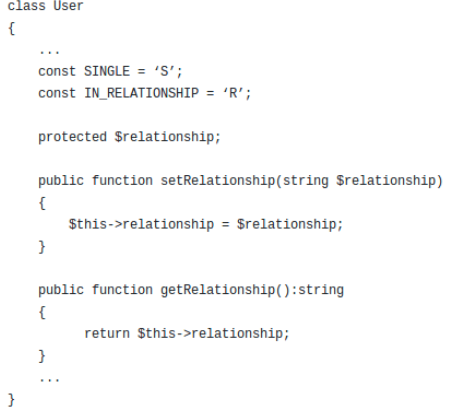
Introduce parameter object

Preserve whole object

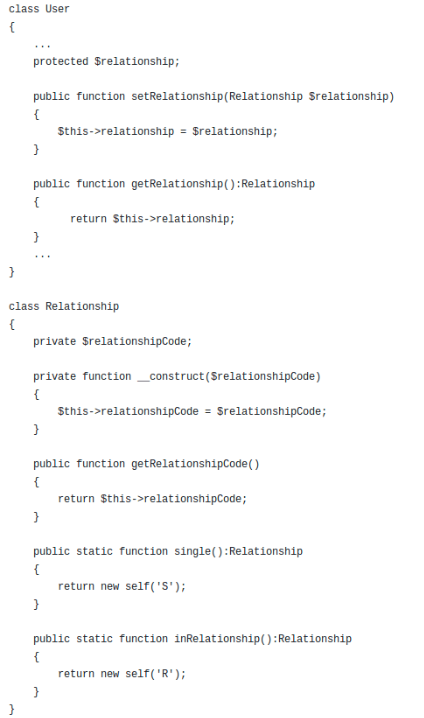
Replace array with object

**Ejemplo:**

**Con mal olor**

****

**Sin mal olor**

****

* **Data Clumps – Grupos de datos**

Se da cuando en un código sus diferentes partes tienen grupos idénticos de variables, tales como parámetros que se conectaran a una base de datos. Este mal olor por lo general se debe a una estructura de programación simple.

**Técnicas de Refactorización**

Extract class

Introduce parameter object

Preserve whole object

**Ejemplo**

**Con mal olor**

public bool SubmitCreditCardOrder ( string primerNombre, string lastName, string de código postal, string streetAddress1, string streetAddress2, string ciudad, string estatal, string país, string phoneNumber, string creditCardNumber, int expirationMonth, int expirationYear, int ImporteVenta)  
{  
// ... Para enviar  
}

**Sin mal olor**

public bool SubmitCreditCardOrder ( **ContactInformation** customerinfo, **la tarjeta de crédito** de la tarjeta, ImporteVenta decimal)  
{  
// ... Para enviar  
}

* **Refused Bequest – Legado rechazado**

Se da cuando una subclase esta usando solo algunos métodos y propiedades heredados de la clase padre, por lo cual la jerarquía estará fuera de lugar. Por esto métodos que no son necesarios no se usarán y emitirán excepciones.

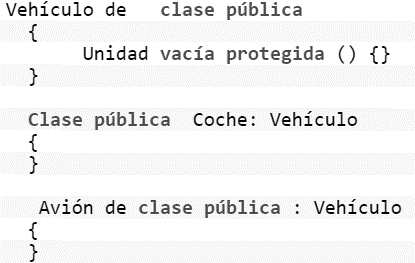
**Técnicas de Refactorización**

Replace inheritance with delegation

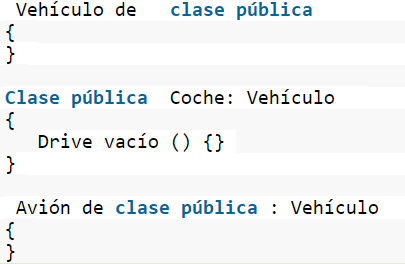
Extract superclass

**Ejemplo:**

**Con mal olor**

****

**Sin mal olor**

****

* **Long Class – Clase Grande**

Este mal olor existe cuando una clase tiene demasiado contenido, es decir tiene muchos métodos/ líneas de código.

**Técnicas de Refactorización**

Extract class

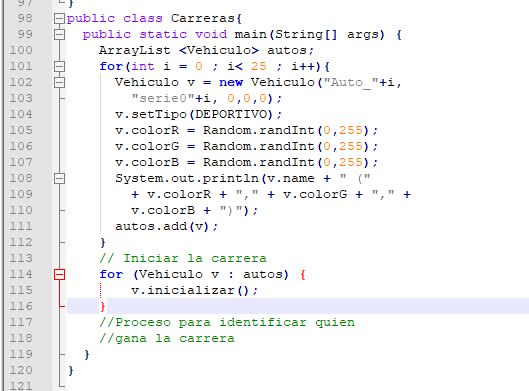
Extract subclass

Extract interface

Duplicate observed data

**Ejemplo:**

**Con mal olor**

****

* **Switch statements – Cambiar declaraciones**

Cuando se tiene un operador switch o una secuencia de if.

**Técnicas de Refactorización**

Extract method

Move method

Replace type code with subclases

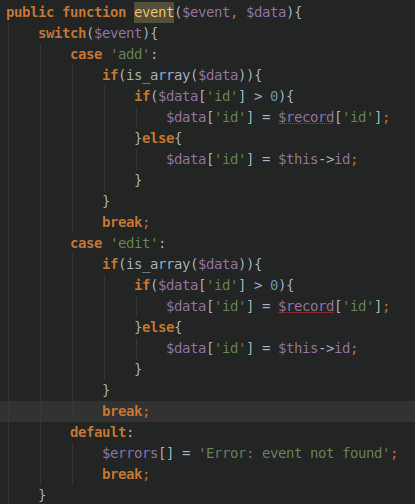
Replace type code with state

Intorduce null object

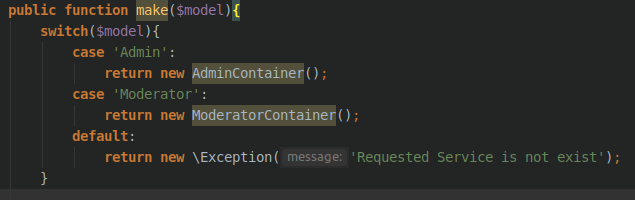
Replace parameter with explicit methods

**Ejemplo:**

**Con mal olor**



**Sin mal olor**



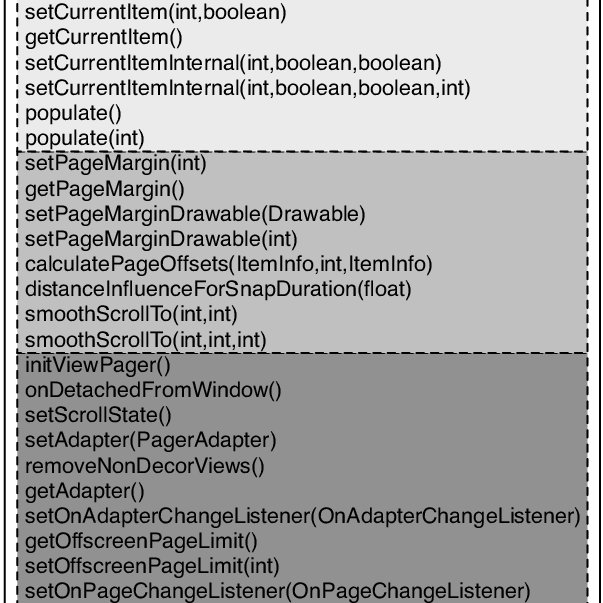
* **Divergent Change – Cambio divergente**

Se da cuando se tiene que realizar muchos cambios en métodos, lo cual provocará que se realicen mas cambios en la clase, desencadenando un conjunto de cambios que se deberán realizar.

**Técnicas de Refactorización**

Extract class

**Ejemplo:**



* **Shotgun Surgery – Cirugía de escopeta**

Este tipo de mal olor realiza diversas modificaciones en un código al momento de realizar operaciones simples como copiar y pegar.

**Técnicas de Refactorización**

Move method

Move field

**Ejemplo:**

**Con mal olor**

Cuenta de clase pública {

       tipo de cadena privada ;

       número de cuenta de cadena privada ;

       cantidad int privada ;

       pública de cuentas (String tipo , Cadena accountNumber , int cantidad )

       {

              esta . cantidad = cantidad ;

              esta . tipo = tipo ;

              esta . accountNumber = accountNumber ;

       }

       pública vacío de débito ( int débito ) lanza Excepción

       {

              si ( cantidad <= 500)

              {

                     lanzar una nueva Excepción ( "El balance mínimo debe ser superior a 500" );

              }

              cantidad = cantidad - débito ;

              Sistema. *out* .println ( "La cantidad actual es" + cantidad );

       }

       transferencia pública nula (Cuenta de , Cuenta a , int cerditAmount ) arroja Excepción

       {

              si ( de . cantidad <= 500)

              {

                     lanzar una nueva Excepción ( "El balance mínimo debe ser superior a 500" );

              }

              a . cantidad = cantidad + cerditAmount ;

       }

       public void sendWarningMessage ()

       {

              si ( cantidad <= 500)

              {

                     Sistema. *out* .println ( "la cantidad debe ser superior a 500" );

              }

       }

}

paquete com.ejemplo.codesmell;

clase pública ShotgunSurgery {

       public static void main (String [] args ) lanza Exception {

              Cuenta acc = cuenta nueva ( "Personal" , "AC1234" , 1000);

              débito acc . (500);

              acc .sendWarningMessage ();

              //acc.debit(500);

       }

}

**Sin mal olor**

paquete com.ejemplo.codesmell;

clase pública AcountRefactored {

       tipo de cadena privada ;

       número de cuenta de cadena privada ;

       cantidad int privada ;

       pública AcountRefactored (String tipo , Cadena accountNumber , int cantidad )

       {

              esta . cantidad = cantidad ;

              esta . tipo = tipo ;

              esta . accountNumber = accountNumber ;

       }

       boolean privado isAccountUnderflow ()

       {

              si ( cantidad <= 500)

              {

                     volver verdadero ;

              }

              devuelve falso ;

       }

       pública vacío de débito ( int débito ) lanza Excepción

       {

              if (isAccountUnderflow ())

              {

                     lanzar una nueva Excepción ( "El balance mínimo debe ser superior a 500" );

              }

              cantidad = cantidad - débito ;

              Sistema. *out* .println ( "La cantidad actual es" + cantidad );

       }

       pública nula transferencia (AcountRefactored de , AcountRefactored a , int cerditAmount ) lanza Excepción

       {

              if (isAccountUnderflow ())

              {

                     lanzar una nueva Excepción ( "El balance mínimo debe ser superior a 500" );

              }

              a . cantidad = cantidad + cerditAmount ;

       }

       public void sendWarningMessage ()

       {

              if (isAccountUnderflow ())

              {

                     Sistema. *out* .println ( "la cantidad debe ser superior a 500" );

              }

       }

}