# BAD SMELLS

**Temporary field (atributo temporal)**: Atributos que se usan solo en algunos métodos y no son esenciales para la clase.

**Data class (clase de datos)**: Clases que solo contienen atributos y métodos de acceso, sin lógica significativa.

**Comments (comentarios excesivos)**: Comentarios innecesarios o en exceso, indicando que el código no es lo suficientemente claro.

**Speculative generality (generalidad especulativa)**: Funcionalidades añadidas "por si acaso" que no son necesarias actualmente.

**Parallel inheritance hierarchies (jerarquías paralelas)**: Jerarquías de clases que crecen en paralelo y requieren cambios coordinados.

**Primitive obsession (obsesión por tipos primitivos)**: Uso excesivo de tipos primitivos para representar entidades más complejas.

**Message chains (cadena de mensajes)**: Llamadas en cadena a métodos a través de varios objetos, generando complejidad.

**Dead code (código muerto)**: Elementos que ya no se usan en el código.

**Duplicated code (código duplicado)**: Código repetido o que realiza la misma tarea en varias partes del sistema.

**Data clumps (grupo de datos)**: Conjuntos de datos que siempre aparecen juntos en múltiples métodos o clases.

**Refused bequest (legado rechazado)**: Subclases que heredan métodos o atributos innecesarios de una clase base.

**Alternative Classes with Different Interfaces (clases alternativas con interfaces diferentes)**: Clases que hacen lo mismo pero con interfaces diferentes, generando inconsistencias.

**Divergent change (cambio divergente)**: Una clase que cambia por diferentes razones, violando el principio de responsabilidad única.

**Large class (clase grande)**: Clases con demasiados atributos o métodos, manejando demasiadas responsabilidades.

**Feature envy (envidia de características)**: Métodos que acceden a los atributos de otra clase más que a los propios.

**Middleman (intermediario)**: Clases que delegan la mayor parte de sus responsabilidades a otras clases sin aportar valor propio.

**Long parameter list (lista de parámetros larga)**: Métodos o constructores que reciben demasiados parámetros.

**Incomplete Library Class (clase de biblioteca incompleta)**: Uso de bibliotecas que no ofrecen toda la funcionalidad necesaria

**Shotgun surgery (cambio en cadena)**: Un pequeño cambio en una clase requiere cambios en varias otras clases.

**Lazy class (clase perezosa)**: Clases que no tienen suficiente funcionalidad para justificar su existencia.

**Switch statement**: Uso excesivo de sentencias switch o if-else anidados.

**Intimidad inapropiada (Inappropriate Intimacy)**: Una clase accede directamente a los atributos privados de otra clase

**Long method (método largo)**: Métodos con demasiadas líneas de código.

# REFACTORIZACIONES

**Add Parameter:** Se aplica cuando un método no tiene suficientes datos para realizar determinadas acciones.

**Pull Up Field:** Se aplica cuando dos clases tiene los mismos atributos.

**Change Value to Reference**: Se aplica cuando se tienen muchas instancias idénticas de una sola clase que necesita reemplazarse con un solo objeto.

**Consolidate Conditional Expression:** Se aplica cuando se tienen múltiples condiciones que conducen al mismo resultado o acción.

**Decompose Conditional**: Se descompone una condición compleja en métodos separados para mayor claridad

**Pull Up Constructor Body:** Se aplica cuando las subclases tienen constructores con código que en su mayoría es idéntico..

**Pull Up Method:** Se aplica cuando las subclases tienen métodos que realizan un trabajo similar.

**Push Down Method:** Se aplica cuando la lógica del método es aplicable solo a ciertas subclases.

**Remove Control Flag**: Se aplica cuando se tiene una variable booleana que actúa como un indicador de control para múltiples expresiones booleanas, en lugar de utilizar la sentencia break, continue o return.

**Remove Parameter:** Se aplica cuando un parámetro no se utiliza en el cuerpo del método.

**Change Bidirectional Association to Unidirectional**: Se aplica cuando se tiene una asociación bidireccional entre clases, pero una de las clases no usa las características de la otra.

**Change Unidirectional Association to Bidirectional:** Se aplica cuando se tiene dos clases y cada una necesita usar las características de la otra, pero la asociación entre ellas es solo unidireccional.

**Collapse Hierarchy:** Se aplica cuando se tiene una jerarquía de clases en la que una subclase es prácticamente igual que su superclase.

**Remove Middleman:** Se aplica cuando una clase tiene demasiados métodos que simplemente delegan a otros objetos.

**Duplicate Observed Data:** Se aplica cuando los datos de dominio se almacenan en clases responsables de la IGU.

**Change Reference to Value:** Se aplica cuando se tiene un objeto de referencia que es demasiado pequeño y que se cambia con poca frecuencia para justificar la gestión de su ciclo de vida.

**Push Down Field:** Se aplica cuando ciertos atributos de la superclase se usan solo en pocas subclases.

**Remove Assignments to Parameters**: Se aplica cuando se asigna algún valor a un parámetro dentro del cuerpo del método.

**Consolidate Duplicate Conditional Fragments:** Se aplica cuando se encuentra un código idéntico en todas las ramas de bloque de condiciones.

**Remove Setting Method**: Se aplica cuando el valor de un atributo se establece solo cuando se crea y no cambia nunca por lo que no tiene sentido mantener el método set.

# REFACTORIZACIONES

**Encapsulate Collection**: Se aplica cuando una clase contiene un atributo de colección y los métodos get y set permiten acceder a la colección completa, en lugar de contar con los métodos para agregar y eliminar elementos de la misma.

**Encapsulate Field**: Se aplica cuando los atributos de una clase son públicos.

**Extract Interface:** Se aplica cuando varios clientes utilizan el mismo subconjunto de la interfaz de una clase o dos clases tienen parte de sus interfaces en común

**Extract Subclass:** Se aplica cuando una clase tiene características que se usan solo en ciertos casos.

**Form Template Method:** Se aplica cuando subclases implementan algoritmos que contienen pasos similares en el mismo orden.

**Rename Method:** Se aplica cuando el nombre del método no explica para qué sirve.

**Replace Array with Object:** Se aplica cuando un vector contiene distintos datos que representan al objeto.

**Replace Constructor with Factory Method**: Se aplica cuando se tiene un constructor complejo que hace algo más que establecer valores de parámetros en atributos del objeto.

**Extract Superclass:** Se aplica cuando se tienen dos clases con campos y métodos comunes.

**Hide Delegate**: Se aplica cuando el cliente obtiene el objeto B de un campo o método del objeto A. Luego, el cliente también llama a un método del objeto B.

**Encapsulate Downcast**: Se aplica cuando un método devuelve un objeto que necesita ser casteado por quienes lo invocan, en lugar de castear dentro de dicho método.

**Extract Class:** Se aplica cuando una clase hace el trabajo de dos.

**Extract Method:** Se aplica cuando se tiene un fragmento de código que se puede agrupar.

**Replace Delegation with Inheritance:** Se aplica cuando una clase contiene muchos métodos simples, que delegan todo a los métodos de otra clase.

**Replace Exception with Test:** Se aplica cuando se lanza una excepción marcada en una condición que podría haberse verificado previamente por quien invoca.

**Rename Field:** Se aplica cudnao el nombre del atributo no explica para qué propósito tiene.

**Replace Conditional with Polymorphism:** Se aplica cuando una estructura condicional realiza varias acciones según el tipo de objeto o propiedades.

**Replace Data Value with Object:** Se aplica cuando una clase (o grupo de clases) contiene un atributo que tiene su propio comportamiento y datos asociados.

**Replace Error Code with Exception:** Se aplica cuando un método devuelve un valor especial que indica un error.

**Replace Inheritance with Delegation:** Se aplica cuando se tiene una subclase que usa solo una parte de los métodos de su superclase (o no es posible heredar datos de la superclase).

# REFACTORIZACIONES

**Hide Method**: Se aplica cuando un método no es usado por otras clases o solo se usa dentro de su propia jerarquía de clases.

**Inline Method**: Se aplica cuando la lógica de un método es tan obvia que no tiene sentido mantener el método.

**Introduce Assertion**: Se aplica para que una parte del código funcione correctamente asumiendo que ciertas condiciones o valores deben ser verdaderos.

**Introduce Foreign Method**: Se aplica cuando una clase de una biblioteca de clases no contiene el método que se necesita y no se tiene la posibilidad de agregarlo. En este caso, habrá que hacer un método que reciba como parámetro un objeto del tipo correspondiente.

**Introduce Null Object:** Se aplica cuando algunos métodos retornan null en lugar de objetos reales y se requiere controlar dichos nulos dentro del código.

**Inline Class**: Se aplica cuando una clase no hace casi nada y no es responsable de nada, y no se planean responsabilidades adicionales para ella.

**Inline Temp:** Se aplica cuando se tiene una variable temporal a la que se le asigna el resultado de una expresión simple y nada más.

**Introduce Explaining Variable:** Se aplica cuando se tiene una expresión complicada.

**Introduce Local Extension:** Se aplica cuando una clase de una biblioteca de clases no contiene alguno de los métodos que se necesitan y no puede agregárselos. En este caso, se deberá heredar de la misma para poder definirlos.

**Introduce Parameter Object:** Se aplica cuando los métodos contienen un grupo repetido de parámetros.

**Replace Magic Number with Symbolic Constant:** Se aplica cuando el código usa un número que tiene cierto significado.

**Replace Nested Conditional with Guard Clauses:** Se aplica cuando se tiene bloque de condiciones anidadas y es difícil determinar el flujo normal de ejecución del código.

**Replace Parameter with Method:** Se aplica cuando un objeto invoca un método y luego pasa el resultado como parámetro a otro método cuando el receptor también podría invocar este método.

**Replace Subclass with Fields:** Se aplica cuando se tiene subclases que difieren solo en sus métodos (de retorno constante).

**Replace Type Code with Class:** Se aplica cuando una clase tiene muchos atributos numéricos para representar distintos tipos y estos no afectan su comportamiento.

**Replace Method with Method Object:** Se aplica cuando se tiene un método largo en el que las variables locales están tan entrelazadas que no puede aplicarse la refactorización "Extract method".

**Replace Parameter with Explicit Methods:** Se aplica cuando un método se divide en partes, cada una de las cuales se ejecuta según el valor de un parámetro.

**Replace Record with Data Class:** Se aplica cuando se necesita interactuar con una estructura de registro en un entorno de programación tradicional.

**Replace Temp with Query:** Se aplica cuando se coloca el resultado de una expresión en una variable local para su uso posterior en el código.

**Replace Type Code with State/Strategy:** Se aplica cuando se tiene un atributo codificado que afecta el comportamiento, pero no se puede usar subclases para deshacerse de él.

# REFACTORIZACIONES

**Move Field:** Se aplica cuando un atributo se usa más en otra clase que en su propia clase.

**Parameterize Method:** Se aplica cuando múltiples métodos realizan acciones similares que son diferentes solo en sus valores internos, números u operaciones.

**Move Method:** Se aplica cuando un método se usa más en otra clase que en su propia clase.

**Preserve Whole Object**: Se aplica cuando se obtienen varios valores de un objeto y luego se pasan como parámetros a un método, en lugar de enviar el objeto completo.

**Replace Type Code with Subclasses**: Se aplica cuando se tiene un atributo codificado que afecta directamente el comportamiento del programa (los valores de este atributo activan distintos códigos en los condicionales).

**Split Temporary Variable:** Se aplica cuando se tiene una variable local que se usa para almacenar varios valores intermedios dentro de un método.

**Self Encapsulate Field:** Se aplica cuando se utiliza el acceso directo a los atributos privados dentro de una clase.

**Substitute Algorithm:** Se aplica cuando se quiere reemplazar un algoritmo por otro más claro.

# BAD SMELLS

**Temporary field (atributo temporal) -->** Extract class. Introduce null object.

**Data class (clase de datos) -->** Move method. Encapsulate field. Encapsulate collection.

**Comments (comentarios excesivos) -->** Extract method. Introduce assertion..

**Speculative generality (generalidad especulativa) -->** Collapse hierarchy. Inline class. Remove parameter. Rename method.

**Parallel inheritance hierarchies (jerarquías paralelas) -->** Move method. Move field

**Primitive obsession (obsesión por tipos primitivos) -->** Replace data value with object. Extract class. Introduce parameter object. Replace array with object. Replace type code with class. Replace type code with subclasses. Replace type code with state/strategy.

**Message chains (cadena de mensajes) -->** Hide delegate.

**Dead code (código muerto) -->** Inline class. Collapse hierarchy. Remove parameter.

**Duplicated code (código duplicado) -->** Extract method. Extract class. Pull up method. Form template method.

**Data clumps (grupo de datos) -->** Extract class. Introduce parameter object. Preserve whole object.

**Refused bequest (legado rechazado) -->** Replace inheritance with delegation.

**Alternative Classes with Different Interfaces (clases alternativas con interfaces diferentes) -->** Rename methods. Move method. Add parameter. Parameterize method. Extract superclass.

**Divergent change (cambio divergente) -->** Extract class.

**Large class (clase grande) -->** Extract class. Extract subclass. Extract interface. Replace data value with object.

**Feature envy (envidia de características) -->** Move method. Move field. Extract method.

**Middleman (intermediario) -->** Remove middleman. Inline method. Replace delegation with inheritance.

**Long parameter list (lista de parámetros larga) -->** Replace parameter with method. Introduce parameter object. Preserve whole object.

**Incomplete Library Class (clase de biblioteca incompleta) -->** Introduce foreign method. Introduce local extension.

**Shotgun surgery (cambio en cadena) -->** Move method. Move field. Inline class.

**Lazy class (clase perezosa) -->** Inline class. Collapse hierarchy

**Switch statement -->** Replace conditional with polymorphism. Replace type code with subclasses. Replace type code with state/strategy. Replace parameter with explicit methods. Introduce null object.

**Intimidad inapropiada (Inappropriate Intimacy) -->** Move method. Move field. Change bidirectional association to unidirectional. Replace inheritance with delegation. Hide delegate.

**Long method (método largo) -->** Extract method. Replace temp with query. Replace method with method object. Decompose conditional.