**Trabajo Autónomo**

Refactoring

1. **Primitive Obsession(obsesión primitiva)**

Este es un code smells el cual se usa variables primitivas, es decir, variables como parámetros los cuales varias variables podrían conforman una solo clase que probablemente maneje validaciones de la misma.

En el código de ejemplo es un problema, ya que, existe una clase en donde ya se puede generar una clase Date que puede manejar las respectivas validaciones, ejemplo seria los 30 días de un mes, y le quitaría la responsabilidad a la clase Persona de validar la fecha de nacimiento.

Técnicas de refactorización: Extract class, Replace Data Value with Object, Replace Type Code with Class, Replace Type, Code with Subclasses or Replace Type Code with State/Strategy.

1. **Switch Statements(Cambiar declaraciones)**

Un code smells muy común al momento hacer código en el cual se tenga condiciones anidadas pudiendo remplazarlo.

Un la clase calcular impuesto se puede observar que hay diferentes tipos de impuestos, esto requería calcular los diferentes tipos de impuestos, sin embargo se puede eliminar este mal olor con la creación de una clase padre Impuesto y que muchos tipos diferentes tipos de impuesto puedan extender de ella, y así también se aplicaría un principio SOLID(Open-Closet principal)

Técnicas de refactorización: Extract method, Replace Conditional with Polymorphism, Replace Parameter with Explicit Methods, Replace Type Code with Subclasses.

1. **Refused Bequest(Legado Rechazado)**

Este mal olor se produce cuando una subclase no utiliza todos los métodos de su clase padre.

En el ejemplo esta la clase Persona que posee varios métodos pagar() y cobrar() son métodos que la clase Cliente y Empleado respectivamente realizan, pero no hacen ambos, si se trata de un ámbito comercial de compra venta. Es por esto que se debe aplicar una técnica de refactorización, ya que, se produce el mal olor “LEGADO RECHAZADO” no utiliza lo que la clase padre entrega a sus clases hijas.

Tecnicas de refactorizacion: Push Down Method, Push Down Field y Replace Inheritance with Delegation.

1. **Data clumps(Grupo de datos)**

Este mal olor se da cuando se puede observar métodos los cuales están cargados de los mismos parámetros de entrada, es decir, como en el método devolver y facturar, hay datos que son fecha, descripción, producto y numero de factura estos parámetros son los mismo en los dos métodos se podría agrupar y formar una nueva clase la cual reduzca el número de parámetros de entradas y de una descripción clara de que recibe el método.

Técnica de refactorización: Extract method, Introduce Parameter Object, Preserve Whole Object

1. **Feature Envy(Característica de envidia)**

Es cuando una clase utiliza métodos muchos métodos de otra clase que sus mismos métodos.

La clase producto está usando los métodos de la clase Calculo y no se encarga de usar los suyos, esto produce un mal olor “Característica envidiada” este se puede corregir implementando métodos en esa clase, es decir, mover el método, a la clase.

Técnica de refactorización: Extract method, move method.