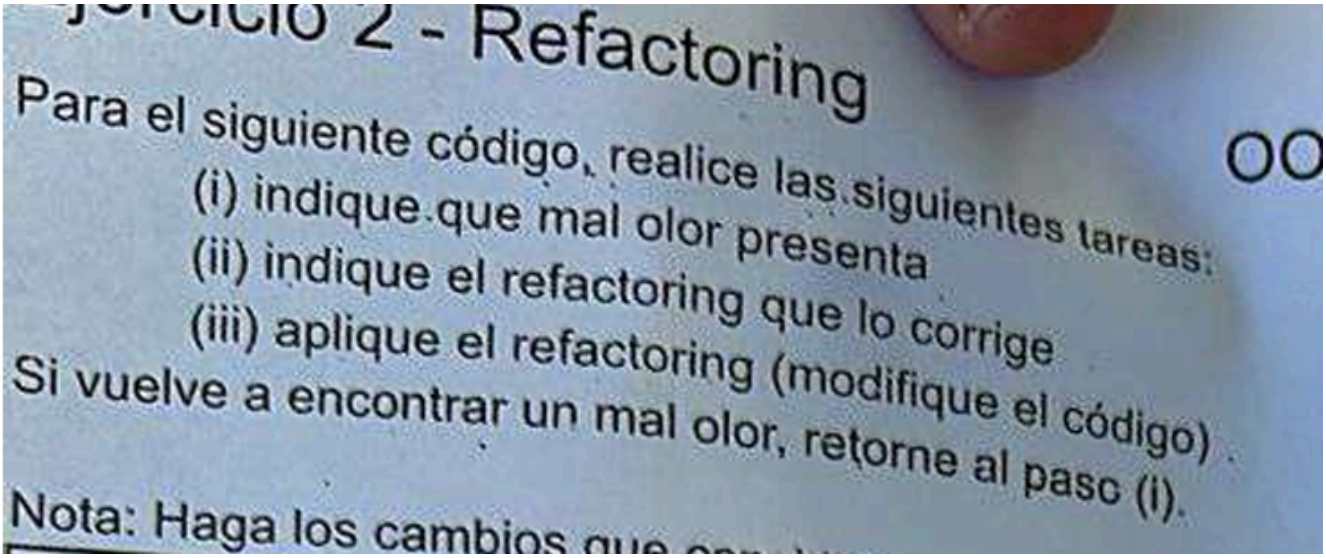
Refactoring



> (1)

Bad smell:

- Codigo duplicado: Se repite la logica para calcular el precio total de los productos.
- Se repite en las lineas 15..17 y 23..25
  Refactoring:
- Extract Method: Extraer el codigo repetido en metodos para luego invocarlos dentro de calcularMontoFinal
  Refactoring Aplicado:

```
public double calcularMontoFinal(){
   double total = 0.0;
   if (this.tipo == 'Efectivo')
   {
    total = calcularPrecioTotal();
     if (total > 100000)
     {
        total = total - DESCUENTO_EFECTIVO;
     }
   }
   else if (this.tipo == 'Tarjeta')
   {
     total = calcularPrecioTotal();
     total = total + Adicional_Tarjeta;
   }
```

```
    return total;
}

private double calcularPrecioTotal()
{
 double total = 0;
 for (Producto producto : this.productos)
 {
    total = total + producto.getPrecio() + (producto.getPrecio() *
producto.getIVA())
 }
 return total;
}
```

> (2)

Bad smell:

- Reinventa la rueda : Utiliza el for para iterar sobre una lista
  Refactoring:
- Replace with pipeline : Reemplazamos el for por la funcion stream que nos proporciona java
  Refactoring Aplicado:

```
private double calcularPrecioTotal()
{
    return this.productos.stream()
       .mapToDouble(p -> p.getPrecio() + (p.getPrecio() * p.getIVA())
       .sum());
}
```

> (3)

Bad smell:

- Switch statement : Se esta preguntando por 'tipos',
- Se pregunta en las lineas 14 y 22.
  Refactoring:
- Replace conditional with polimorphysm:
  Pasos:
- Crear una clase abstracta 'TipoDePago' con metodo abstracto para modificar el total del pago.

- Crear dos subclases concretas: Clase 'Efectivo' y clase 'Tarjeta', que extiendan la clase abstracta 'TipoDePago'
- Mover datos a subclases (las variables estaticas de DESCUENTO_EFECTIVO y ADICIONAL_TARJETA)
- Se eliminan los condicionales
- Se elimina la variable 'tipo' haciendo referencia al metodo de pago
  - Se agrega una variable de tipo TipoDePago
  Refactoring Aplicado:

```
public abstract class TipoDePago
{
   public abstract double modificarPrecioTotal(double precioTotal);
}

public class Efectivo extends TipoDePago
{
    private static final double DESCUENTO_EFECTIVO = 2000.0;
    @Override
    public double modificarPrecioTotal(double precioTotal)
    {
        if (precioTotal > 1000000) precioTotal = precioTotal -
DESCUENTO_EFECTIVO;
        return precioTotal;
    }
}

public class Tarjeta extends TipoDePago
{
    private static final double ADICIONAL_TARJETA = 1000.0;
    @Override
    public double modificarPrecioTotal(double precioTotal)
    {
        return precioTotal + ADICIONAL_TARJETA;
    }
}

public class Pago {
    private List<Producto> productos;
    private TipoDePago tipoPago;
    public Pago(List<Producto> productos, TipoDePago tipoPago)
    {
        this.productos = productos;
        this.tipoPago = tipoPago;
    }
```

```
    public double calcularMontoFinal()
    {
        double total = this.calcularPrecioTotal();
        total = tipoPago.modificarPrecioTotal(total)
        return total;
    }

    private double calcularPrecioTotal()
    {
        return this.productos.stream()
           .mapToDouble(p -> p.getPrecio + (p.getPrecio * p.getIVA())
           .sum());
    }
}
```

> (4)

Bad smell:

- Feature envy : Envidia de atributos en la linea de (producto.getPrecio * producto.getIVA)
- La clase Producto debe ser la encargada de devolver el precio del producto con IVA
  Refactoring:
- Extract method: Se debe extraer la parte donde se produce la envidia de atributos
  - Se crea el metodo con nombre 'getPrecioConIVA'
- Move method: Se mueve el metodo de la clase Pago, hacia la clase producto
  Refactoring Aplicado:

```
public class Producto {
   private double precio;
   private String IVA;

   public Producto (double precio, double IVA)
   {
       this.precio = precio;
       this.IVA = IVA;
   }

    public double getPrecioConIVA()
    {
       return this.precio + (this.precio * this.IVA)
    }
}
```

```
public class Pago {
// .......

    private double calcularPrecioTotal()
    {
        return this.productos.stream()
           .mapToDouble(p -> p.getPrecioConIVA()
           .sum());
    }
}
```