Actividad 6.1 – Code Smells y Refactorización

Clase `Animal`

Code Smells detectados:

- Método estático depende de instancia (`tipo`)**: `tipo` es un atributo de instancia, pero se usa en un método `static` .
- `switch` o `if` **largo (Code Smell: *Large Conditional*)**
- **Falta de polimorfismo (Code Smell: *Type Checking*)**
- **Variable no declarada (`resultado`)**

Refactorización :

- crear una jerarquía para cada tipo de animal.
- Eliminar uso de condicionales.
- Eliminar método estático.

```
public abstract class Animal {
    private String nombre;
    private int edad;

    public Animal(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public abstract String sonido();
}

public class Vaca extends Animal {
    public Vaca(String nombre, int edad) {
        super(nombre, edad);
    }

    @Override
    public String sonido() {
        return "Muu";
    }
}
```

Clases `FormaDePagoTarjeta` y `FormaDePagoPaypal`

Code Smells detectados:

- **Duplicación de lógica de pagos (Code Smell: *Duplicated Code*)**
- **Falta de abstracción común (Code Smell: *Speculative Generality*)**

Refactorización

- Crear una **interfaz común** `FormaDePago` .
- Unificar nombres de métodos (`usar` y `hacerPago`).

```
public interface FormaDePago {
    void pagar(double cantidad);
}

public class FormaDePagoTarjeta implements FormaDePago {
    private String numeroTarjeta;
    private String fechaCaducidad;
    private String cvv;
```

```
    public FormaDePagoTarjeta(String numeroTarjeta, String fechaCaducidad, String cvv) {
        this.numeroTarjeta = numeroTarjeta;
        this.fechaCaducidad = fechaCaducidad;
        this.cvv = cvv;
    }

    @Override
    public void pagar(double cantidad) {

    }
}

public class FormaDePagoPaypal implements FormaDePago {
    private String email;
```

`CuentaBancaria` y `Banco`

Code Smells detectados:

- Falta de encapsulamiento de lógica (Code Smell: *Feature Envy*)**: `Banco` calcula intereses con lógica que debería estar en `CuentaBancaria`.

Refactorización:

- Mover la lógica del cálculo del interés a `CuentaBancaria`.

```
public class CuentaBancaria {
    private double balance;
    private double interes;

    public double calcularInteres(int anyos) {
        return balance * interes * anyos;
    }
}

public class Banco {
    private List<CuentaBancaria> cuentas;

    public double calcularInteresTotal(int anyos) {
        double total = 0;
        for (CuentaBancaria cuenta : cuentas) {
            total += cuenta.calcularInteres(anyos);
        }
        return total;
    }
}
```

Clase `Pedido`

Code Smells detectados:

- **Método muy largo (Code Smell: *Long Method*)**
- **Demasiadas responsabilidades (Code Smell: *God Class*)**
- **Nombres ambiguos (`formaDePago` como `String` en lugar de objeto)**
- **Duplicación de lógica de descuento y gastos de envío**

Refactorización propuesta:

- Extraer métodos para calcular descuentos, gastos de envío, etc.
- Usar clase para `FormaDePago`.
- Posiblemente aplicar **Patrón Strategy**.

Código refactorizado (fragmento):

```java
public class Pedido {
    private List<Item> items;
    private FormaDePago formaDePago;

    public double calcularTotal() {
        double total = calcularSubtotal();
        total += calcularGastosEnvio();

        if (formaDePago instanceof Contrareembolso) {
            total += 5;
        }

        if (total > 100) {
            total *= 0.9;
        }

        return total;
    }

    private double calcularSubtotal() {
        double subtotal = 0;
        for (Item item : items) {
            subtotal += item.isRebajado() ? item.getPrecio() * 0.85 : item.getPrecio();
```