PATRÓN DE DISEÑO: PROTOTIPO

Propósito: El patrón de diseño Prototype es un patrón creacional que permite crear nuevos objetos copiando instancias existentes, en lugar de crear nuevas instancias desde cero. Esto es útil cuando la creación de un objeto es costosa o compleja.

Implementación en el Proyecto:

Tenemos la clase Metododepago.java dentro del paquete Vista. Esta clase representa la lógica del proceso de pago en el sistema, incluyendo atributos como el total de la venta, los métodos de pago seleccionados y los montos de pago.

Para aplicar el patrón Prototype, primero necesitamos una interfaz que establezca el contrato para la clonación de objetos.

Definición de la interfaz Prototype:

```
public interface PagoPrototype {
    Metododepago clonar();
}
```

Esta interfaz define el método `clonar()`, que cada clase que implemente el patrón debe definir para devolver una copia de sí misma.

Implementación en la Clase Metododepago:

```
public class Metododepago extends javax.swing.JFrame implements PagoPrototype {
    private double totalVenta;
    private double pago1;
    private double pago2;
    private String metodoPago1;
    private String metodoPago2;


    public Metododepago(double totalVenta, double pago1, double pago2, String
metodoPago1, String metodoPago2) {
        this.totalVenta = totalVenta;
        this.pago1 = pago1;
        this.pago2 = pago2;
        this.metodoPago1 = metodoPago1;
        this.metodoPago2 = metodoPago2;
    }
```

```
    @Override
    public Metododepago clonar() {
        return new Metododepago(this.totalVenta, this.pago1, this.pago2, this.metodoPago1,
this.metodoPago2);
    }
}
```

Esto garantiza que cada instancia de `Metododepago` pueda clonarse sin necesidad de crear una nueva desde cero.

Uso del Patrón Prototype en la Lógica del Programa:

```
// Crear el objeto original del pago
Metododepago pagoOriginal = new Metododepago(totalVenta, pago1, pago2, metodopago1,
metodopago2);


// Clonar el pago original
Metododepago pagoClonado = pagoOriginal.clonar();


// Modificar el clon sin afectar el original
pagoClonado.setPago1(50.0);


// Verificar que el clon es independiente
System.out.println("Pago Original:");
pagoOriginal.imprimirDatos();
System.out.println("\nPago Clonado:");
pagoClonado.imprimirDatos();
```

Estructura del Patrón:

1. **Interfaz Prototype:** Define el método `clonar()`.
2. **Clase Concreta:** Implementa la interfaz y define la lógica de clonación.
3. **Uso en el Programa:** Se crea un objeto, se clona y se modifican los atributos del clon sin afectar el original.

Pros y contras:

Ventajas:

- Permite crear nuevos objetos de forma eficiente.
- Reduce la necesidad de crear nuevas instancias desde cero.
- Facilita la creación de copias sin afectar el objeto original.

Desventajas:

- Puede aumentar la complejidad si el objeto tiene referencias a otros objetos que también deben clonarse.
- En algunos casos, una clonación profunda puede ser más compleja de implementar.

El patrón Prototype es útil en sistemas donde la creación de objetos es costosa y se necesita una forma eficiente de generar copias sin modificar el objeto original.