Tecnológico Nacional De México

Instituto Tecnológico de Oaxaca

Ingeniería En Sistemas Computacionales

Diseño e Implementación de Software con Patrones.

7 am – 8 am

Unidad 2

“Patrón de Diseño Prototype”

Presenta:

Copy Max

Nombres	Numero de Control
Bautista Fabian Max	**C19160532**
Celis Delgado Jorge Eduardo	**21160599**
Flores Guzmán Alan Ismael	**20161193**
García Osorio Bolívar	**20161819**
Pérez Barrios Diego	**21160750**
Perez Martínez Edith Esmeralda	**21160752**
Sixto Morales Ángel	**21160797**

Periodo Escolar:

Febrero – Julio

Grupo:

7SB

Maestro:

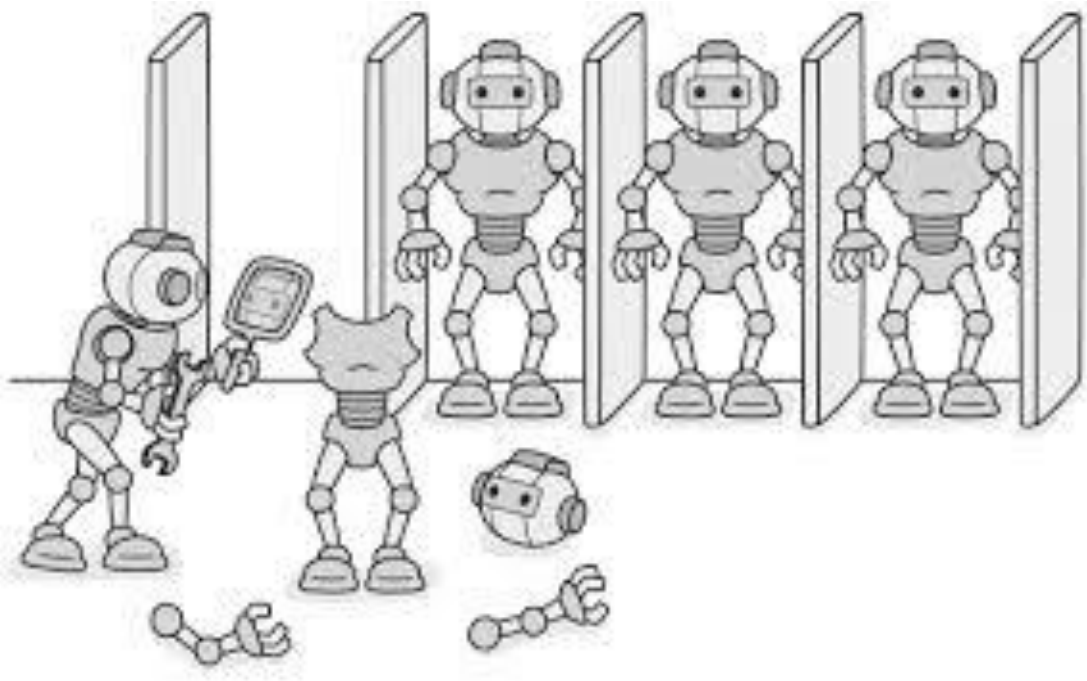
Espinoza Pérez Jacob

Oaxaca de Juárez, Oaxaca **Marzo, 2025**

INDICE

Patrón de Diseño Prototype

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:

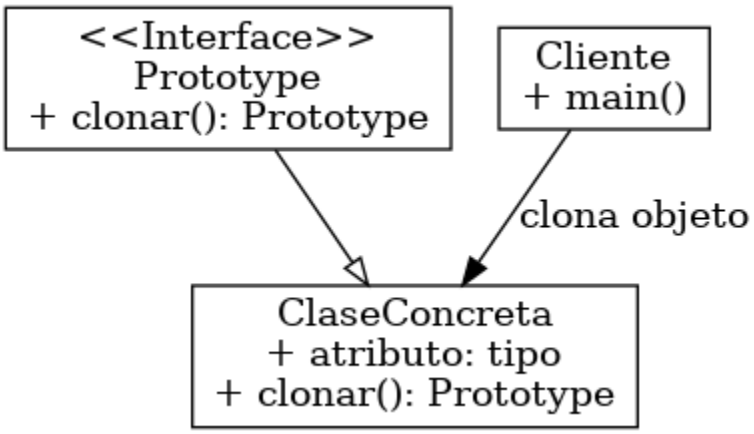
```
public class Metododepago extends javax.swing.JFrame implements PagoPrototype {

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 UML



Ventajas y Desventajas

Ventajas	Desventajas
Permite crear nuevos objetos de forma eficiente.	Puede aumentar la complejidad si el objeto tiene referencias a otros objetos que también deben clonarse.
Reduce la necesidad de crear nuevas instancias desde cero.	En algunos casos, una clonación profunda puede ser más compleja de implementar.

Conclusión

El patrón de diseño Prototype es una solución eficiente para la creación de objetos al permitir su clonación en lugar de instanciarlos desde cero. Esto resulta especialmente útil cuando la creación de objetos es costosa en términos de recursos o tiempo. Además, este patrón ayuda a reducir la dependencia de las clases concretas y fomenta la flexibilidad en el diseño del software.