**Tecnológico Nacional De México**

**Instituto Tecnológico de Oaxaca**

**Ingeniería En Sistemas Computacionales**

**Diseño e Implementación de Software con Patrones.**

7 am – 8 am

**Unidad 3**

**“**Patrón de Diseño Proxy”

Presenta:

**Implementación de los Patrones en el Sistema 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** |
| Pérez Martínez Edith Esmeralda | **21160752** |
| Sixto Morales Ángel | **21160797** |

Periodo Escolar:

**Febrero – Julio**

Grupo:

**7SB**

Maestro:

**Espinoza Pérez Jacob**

**Oaxaca de Juárez, Oaxaca** **Abril 2025**

# INDICE

# INTRODUCCIÓN

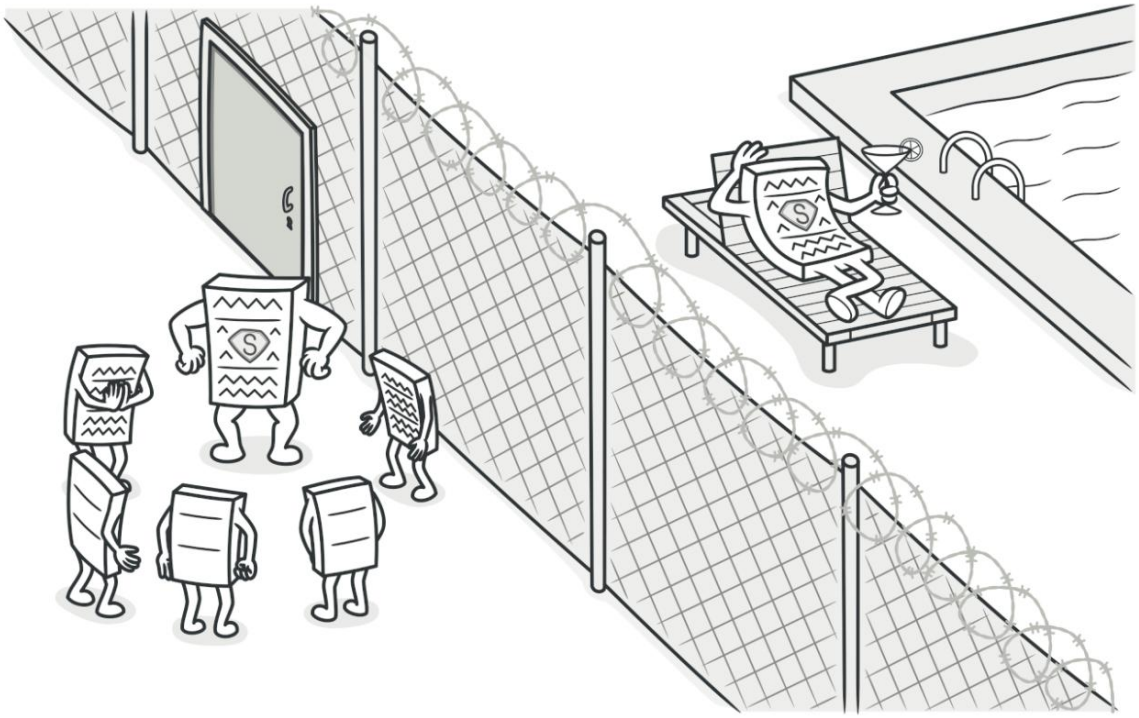
El patrón de diseño Proxy se presenta como una solución eficaz para situaciones donde se necesita gestionar el acceso a objetos complejos o de alto costo computacional. Es comúnmente utilizado en contextos como la carga diferida de recursos, el control de acceso a servicios remotos, o la implementación de cachés. Mediante la creación de un objeto "sustituto", el patrón Proxy permite interceptar y modificar las solicitudes dirigidas al objeto real, brindando así una capa de control adicional que mejora el rendimiento, la seguridad o la eficiencia del sistema.

Este patrón actúa como un intermediario entre el sistema principal y el recurso objetivo, permitiendo controlar el acceso, aplicar validaciones o incluso reducir el uso innecesario de recursos.

Por ejemplo, ciertas operaciones como la generación de reportes, la visualización de información confidencial o incluso la conexión a servicios externos pueden beneficiarse del uso del patrón de diseño Proxy.

# PATRÓN DE DISEÑO PROXY:  IMPLEMENTADO EN COPY MAX

Proxy es un patrón de diseño estructural que te permite proporcionar un sustituto o marcador de posición para otro objeto. Un proxy controla el acceso al objeto original, permitiéndote hacer algo antes o después de que la solicitud llegue al objeto original.



En el sistema de gestión de clientes desarrollado, el patrón Proxy se implementó para restringir el acceso a ciertas funcionalidades (modificar, eliminar y exportar clientes) dependiendo del rol del usuario (Administrador o Empleado).

Cada funcionalidad restringida cuenta con una interfaz (como Modificador, Eliminador, Exportador) que define el método permitido. Luego, se implementa una clase 'real' que realiza la acción si el usuario tiene permiso, y una clase 'proxy' que verifica si el usuario es Administrador antes de permitir el acceso.

# CÓDIGO IMPLEMENTADO

- Modificador: Es la interfaz que define el método modificarCliente(Clientesclass).
- ModificadorReal implementa la lógica real para modificar un cliente.
- ModificadorProxy también implementa la interfaz, pero actúa como un filtro, verificando si el usuario tiene permisos antes de permitir que se ejecute la operación real.

El patrón aquí se usa para controlar el acceso a una operación sensible (modificar datos de cliente) según los permisos.

## Interfaz Modificador.java

```
package Modelo;

import Modelo.Clientesclass;

public interface Modificador {
   String modificarCliente(Clientesclass cliente);
}
```

---

## Clase ModificadorReal.java

```
package Modelo;

public class ModificadorReal implements Modificador {

   @Override
   public String modificarCliente(Clientesclass cliente) {
      cliente.actualizarClienteBD(cliente);
      return "Cliente modificado correctamente.";
   }
}
```

---

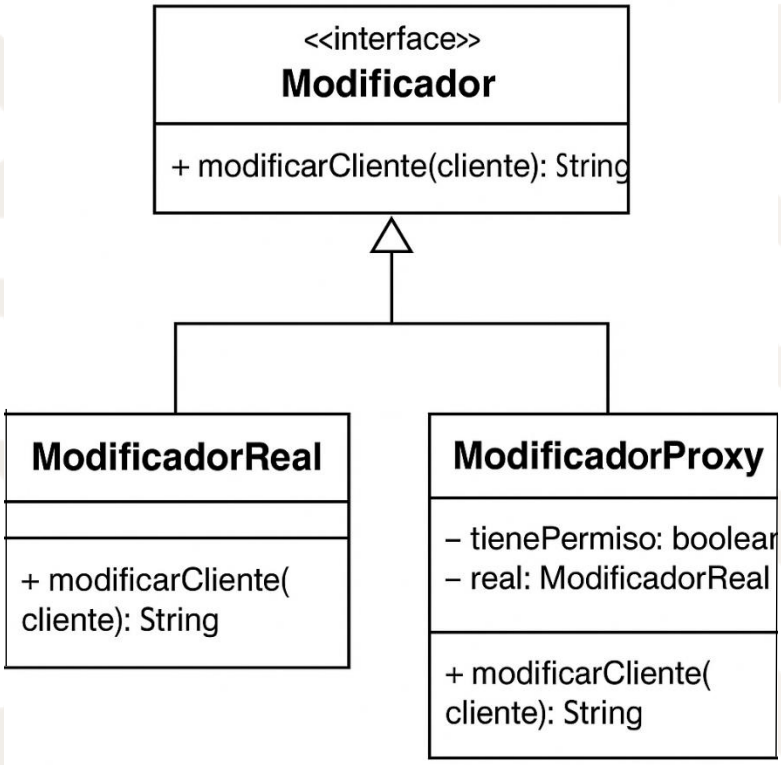
## Clase ModificadorProxy.java

```
package Modelo;

public class ModificadorProxy implements Modificador {
   private final boolean tienePermiso;
   private final ModificadorReal real;

   public ModificadorProxy(boolean tienePermiso) {
      this.tienePermiso = tienePermiso;
      this.real = new ModificadorReal();
   }

   @Override
   public String modificarCliente(Clientesclass cliente) {
      if (!tienePermiso) {
         return "Acceso denegado: No tienes permisos para modificar
clientes.";
      }
      return real.modificarCliente(cliente);
   }
}
```

---

Cuando un usuario con rol 'Empleado' intenta modificar, eliminar o exportar un cliente, se le muestra un mensaje que indica que no tiene permisos para realizar esa acción. Esto garantiza la seguridad del sistema y demuestra el correcto uso del patrón Proxy.

# ESTRUCTURA UML




# VENTAJAS Y DESVENTAJAS

| Ventajas | Desventajas |
|---|---|
| Control de acceso: Permite restringir operaciones según permisos del usuario. | Mayor complejidad: Añade una capa extra de abstracción al sistema. |
| Modularidad: Separa la lógica de control de acceso de la lógica principal. | Rendimiento: Llamar al proxy puede introducir una leve sobrecarga. |
| Extensibilidad: Es fácil agregar validaciones o funciones adicionales sin modificar el objeto real. | Dependencia del proxy: Un mal diseño puede hacer que todo dependa demasiado del proxy. |

# CONCLUSIÓN

El patrón de diseño Proxy representa una solución eficiente y elegante para controlar el acceso a recursos sensibles dentro de un sistema. Su implementación permite garantizar que solo los usuarios autorizados puedan modificar la información de los clientes, fortaleciendo así la seguridad y la integridad de los datos.