TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE OAXACA

Carrera:

Ingeniería en Sistemas Computacionales

Materia:

Diseño e Implementación de software con patrones

Documentación de patrón Strategy

Docente:

Espinoza Pérez Jacob

Equipo:

Ordaz Pacheco Ruudvan

Santos Manuel Julia Marlenny

Vera Acevedo Héctor Aramís

Grupo:

7SB

Fecha de entrega:

11/05/2025

Documentación del Patrón Strategy en el Sistema de Ventas

Introducción

Este documento detalla la implementación del **patrón Strategy** en un sistema de ventas para manejar diferentes estrategias de cálculo de IVA. El objetivo es crear un diseño flexible que permita:

- **Cambiar fácilmente** entre diferentes porcentajes de IVA.
- **Mantener limpio y escalable** el código.
- **Evitar modificaciones constantes** en la lógica principal cuando se añadan nuevos tipos de IVA.

Implementación del Patrón Strategy

- **Interfaz** IvaStrategy

Define el contrato que todas las estrategias de IVA deben seguir.

```
public interface IvaStrategy {
    double calcularIva(double precio);
    String getDescripcion();
    int getPorcentaje(); // Nuevo método
}
```

Implementan la interfaz para diferentes porcentajes de IVA.

IvaCero.java **(0% IVA)**

```
*/
public class IvaCero implements IvaStrategy {
    @Override
    public double calcularIva(double precio) {
        return 0;
    }

    @Override
    public String getDescripcion() {
        return "No grava IVA";
    }

    public int getPorcentaje() {
    return 0;
}
}
```

IvaDoce.java **(12% IVA)**

```
public class IvaDoce implements IvaStrategy {
    @Override
    public double calcularIva(double precio) {
        return precio * 0.12;
    }

    @Override
    public String getDescripcion() {
        return "12%";
    }

    @Override
public int getPorcentaje() {
    return 12;
}

}
```

IvaDieciseis.java **(16% IVA)**

```
public class IvaDieciseis implements IvaStrategy {
    @Override
    public double calcularIva(double precio) {
        return precio * 0.16;
    }

    @Override
    public String getDescripcion() {
        return "16%";
    }

    @Override
public int getPorcentaje() {
    return 16;
}


}
```

Clase Producto **(modelo)**

Adaptada para usar el patrón Strategy.

```
import controlador.IvaCero;
import controlador.IvaStrategy;


public class Producto {

    //Atributos
    private int idProducto;
    private String nombre;
    private int cantidad;
    private double precio;
    private String descripcion;
    private IvaStrategy ivaStrategy;
    private int idCategoria;
    private int estado;

    //Contructor
    public Producto(){
        this.idProducto = 0;
        this.nombre = "";
        this.cantidad = 0;
        this.precio = 0.0;
        this.descripcion = "";
        this.ivaStrategy = new controlador.IvaCero(); // Valor por defecto
        this.idCategoria = 0;
        this.estado = 0;
    }
    //Contructor sobrecargado
    public Producto(int idProducto, String nombre, int cantidad, double precio, String descripcion, int idCategoria, int estado) {
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.cantidad = cantidad;
        this.precio = precio;
        this.descripcion = descripcion;
        this.ivaStrategy = new controlador.IvaCero();
        this.idCategoria = idCategoria;
        this.estado = estado;
    }
```

```java
public void setIvaStrategy(IvaStrategy ivaStrategy) {
        this.ivaStrategy = ivaStrategy;
    }

    public IvaStrategy getIvaStrategy() {  // Añadimos este método getter
        return this.ivaStrategy;
    }

    public double calcularIva() {
        return ivaStrategy.calcularIva(this.precio);
    }

    public double getPrecioConIva() {
        return this.precio + calcularIva();
    }
```

Controlador Ctrl_Producto

Modificado para guardar el porcentaje de IVA en la base de datos.

```java
public boolean actualizar(Producto objeto, int idProducto) {
    boolean respuesta = false;
    Connection cn = Conexion.conectar();
    try {

        PreparedStatement consulta = cn.prepareStatement("update tb_producto set nombre=?, cantidad = ?, precio = ?, descri
        consulta.setString(1, objeto.getNombre());
        consulta.setInt(2, objeto.getCantidad());
        consulta.setDouble(3, objeto.getPrecio());
        consulta.setString(4, objeto.getDescripcion());
        consulta.setInt(6, objeto.getIvaStrategy().getPorcentaje());
        consulta.setInt(6, objeto.getIdCategoria());
        consulta.setInt(7, objeto.getEstado());

        if (consulta.executeUpdate() > 0) {
            respuesta = true;
        }
        cn.close();
    } catch (SQLException e) {
        System.out.println("Error al actualizar producto: " + e);
    }
    return respuesta;
}
```

```java
public boolean guardar(Producto objeto) {
boolean respuesta = false;
Connection cn = Conexion.conectar();
try {
    PreparedStatement consulta = cn.prepareStatement("insert into tb_producto values(?,?,?,?,?,?,?,?)");
    consulta.setInt(1, 0);//id
    consulta.setString(2, objeto.getNombre());
    consulta.setInt(3, objeto.getCantidad());
    consulta.setDouble(4, objeto.getPrecio());
    consulta.setString(5, objeto.getDescripcion());
    // Cambiamos para usar getIvaStrategy()
    consulta.setInt(6, objeto.getIvaStrategy().getPorcentaje());// Guardamos la descripción
    consulta.setInt(7, objeto.getIdCategoria());
    consulta.setInt(8, objeto.getEstado());

    if (consulta.executeUpdate() > 0) {
        respuesta = true;
    }
    cn.close();
} catch (SQLException e) {
    System.out.println("Error al guardar producto: " + e);
}
return respuesta;
```

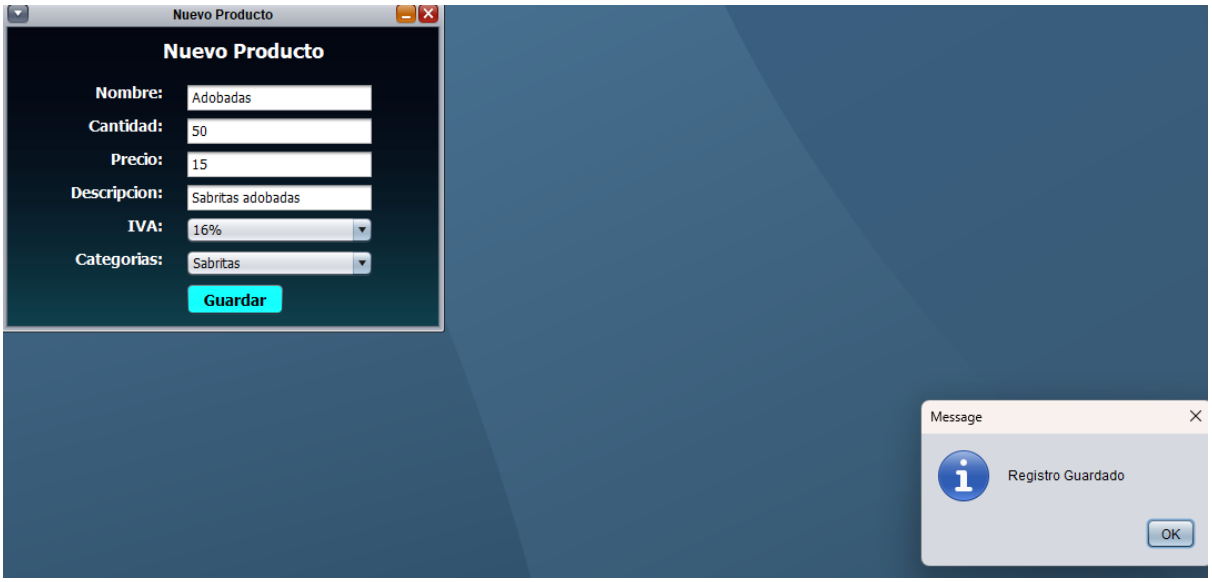
Interfaz Gráfica InterProducto **(vista)**

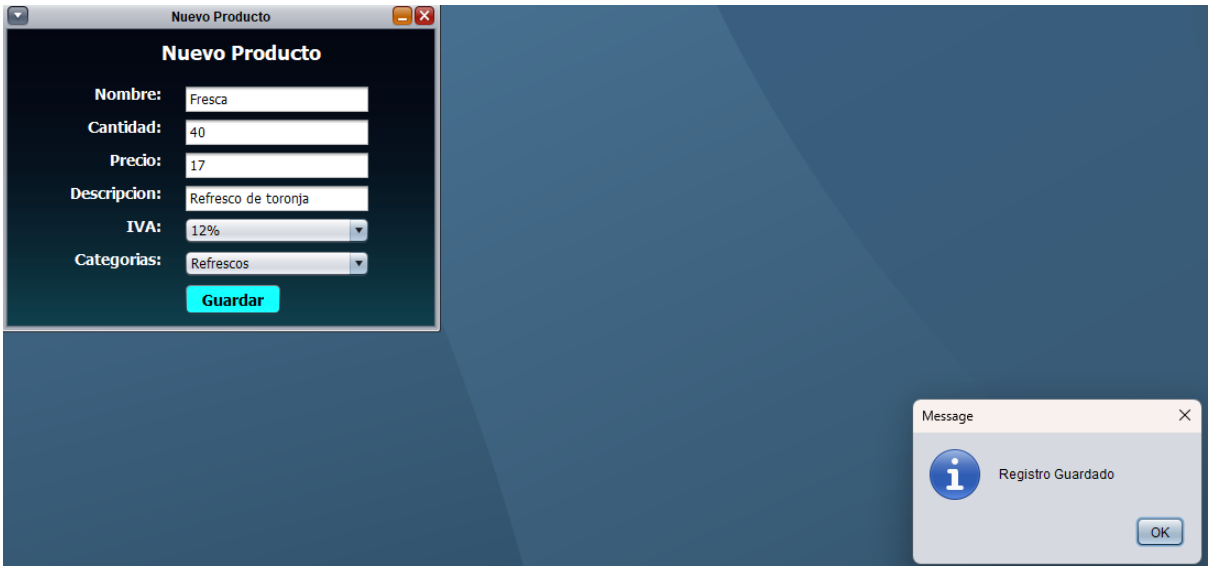
Adaptada para seleccionar la estrategia de IVA.

```java
        Producto producto = new Producto();
        Ctrl_Producto controlProducto = new Ctrl_Producto();
        String iva = "";
        String categoria = "";
        String ivaSeleccionado = jComboBox_iva.getSelectedItem().toString().trim();
    IvaStrategy estrategiaIva;
        categoria = jComboBox_categoria.getSelectedItem().toString().trim();

        //validar campos
```

```java
                                producto.setDescripcion(txt_descripcion.getText().trim());
                                //Porcentaje IVA
                               switch(ivaSeleccionado) {
        case "No grava iva":
            estrategiaIva = new controlador.IvaCero(); // Referencia completa
            break;
        case "12%":
            estrategiaIva = new controlador.IvaDoce();
            break;
        case "16%":
            estrategiaIva = new controlador.IvaDieciseis();
            break;
        default:
            estrategiaIva = new controlador.IvaCero();
    }

    producto.setIvaStrategy(estrategiaIva);
```

Resultados





42	Fresca	40	17.0	Refresco de...	2.04	Refrescos	1
41	Adobadas	50	15.0	Sabritas ad...	0.0	Sabritas	1

Conclusión

La implementación del **patrón Strategy** en el sistema de ventas para el cálculo de IVA no solo resuelve un problema técnico específico, sino que también introduce una serie de beneficios significativos que mejoran la calidad, escalabilidad y mantenibilidad del software a largo plazo. A continuación, se detallan los aspectos más relevantes de esta implementación:

- **Flexibilidad:** Cambiar el cálculo de IVA sin modificar la clase Producto.
- **Escalabilidad:** Añadir nuevos tipos de IVA fácilmente.
- **Mantenibilidad:** Código más limpio y organizado.