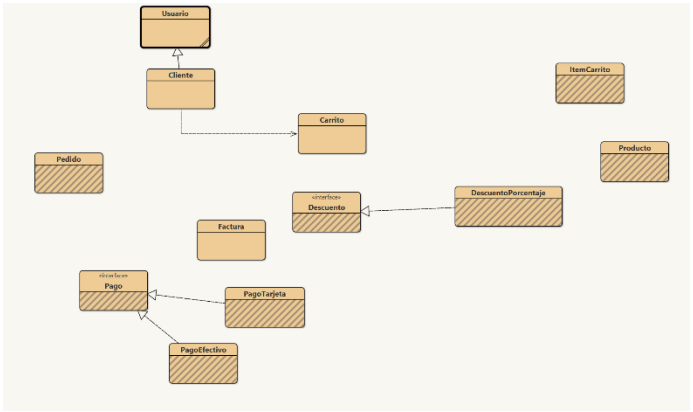
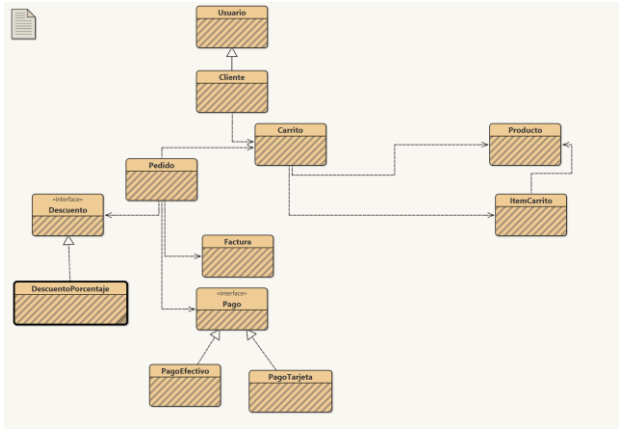
Antes



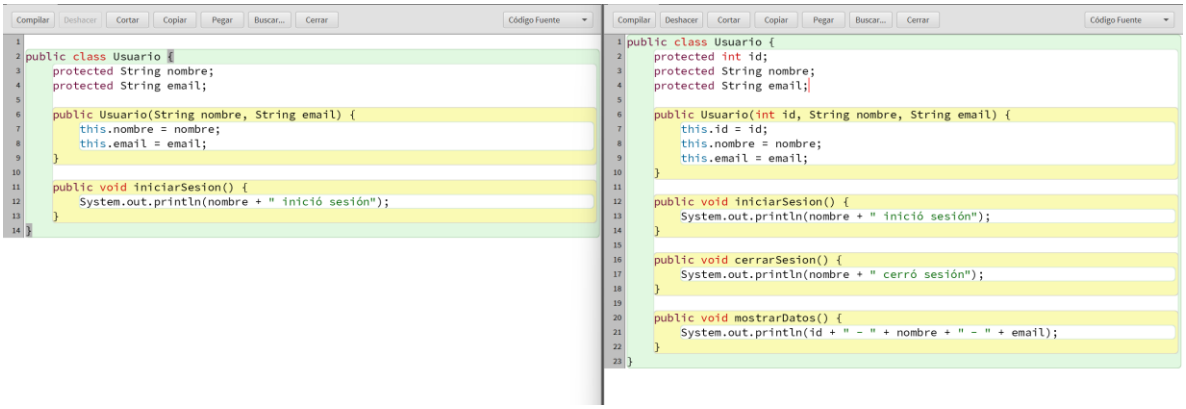
Después



Mejoras realizadas

En la primera versión solo se tenia las clases ideadas y conexiones entre ciertas clases e interfaces, en la mejora se agregaron los atributos, métodos además que se organizo para cumplir con la mayoría de los principios Solid

Usuario



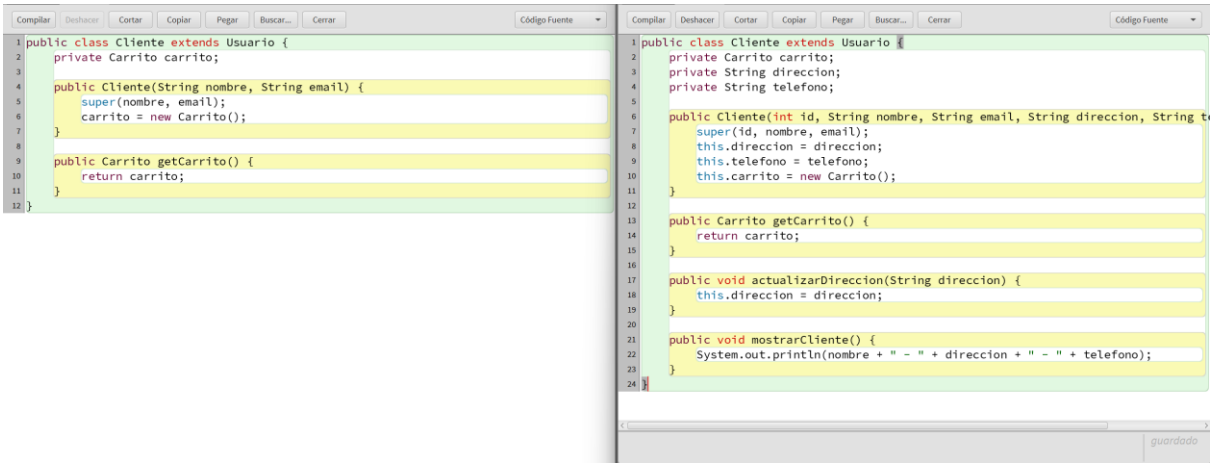
```java
public class Usuario {
    protected String nombre;
    protected String email;

    public Usuario(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }

    public void iniciarSesion() {
        System.out.println(nombre + " inició sesión");
    }
}
```

```java
public class Usuario {
    protected int id;
    protected String nombre;
    protected String email;

    public Usuario(int id, String nombre, String email) {
        this.id = id;
        this.nombre = nombre;
        this.email = email;
    }

    public void iniciarSesion() {
        System.out.println(nombre + " inició sesión");
    }

    public void cerrarSesion() {
        System.out.println(nombre + " cerró sesión");
    }

    public void mostrarDatos() {
        System.out.println(id + " - " + nombre + " - " + email);
    }
}
```

SRP

La clase Usuario tiene como única responsabilidad gestionar la información y acciones básicas de un usuario (id, nombre, email, iniciar/cerrar sesión).

Cliente



```java
public class Cliente extends Usuario {
    private Carrito carrito;

    public Cliente(String nombre, String email) {
        super(nombre, email);
        carrito = new Carrito();
    }

    public Carrito getCarrito() {
        return carrito;
    }
}
```

```java
public class Cliente extends Usuario {
    private Carrito carrito;
    private String direccion;
    private String telefono;

    public Cliente(int id, String nombre, String email, String direccion, String t
        super(id, nombre, email);
        this.direccion = direccion;
        this.telefono = telefono;
        this.carrito = new Carrito();
    }

    public Carrito getCarrito() {
        return carrito;
    }

    public void actualizarDireccion(String direccion) {
        this.direccion = direccion;
    }

    public void mostrarCliente() {
        System.out.println(nombre + " - " + direccion + " - " + telefono);
    }
}
```

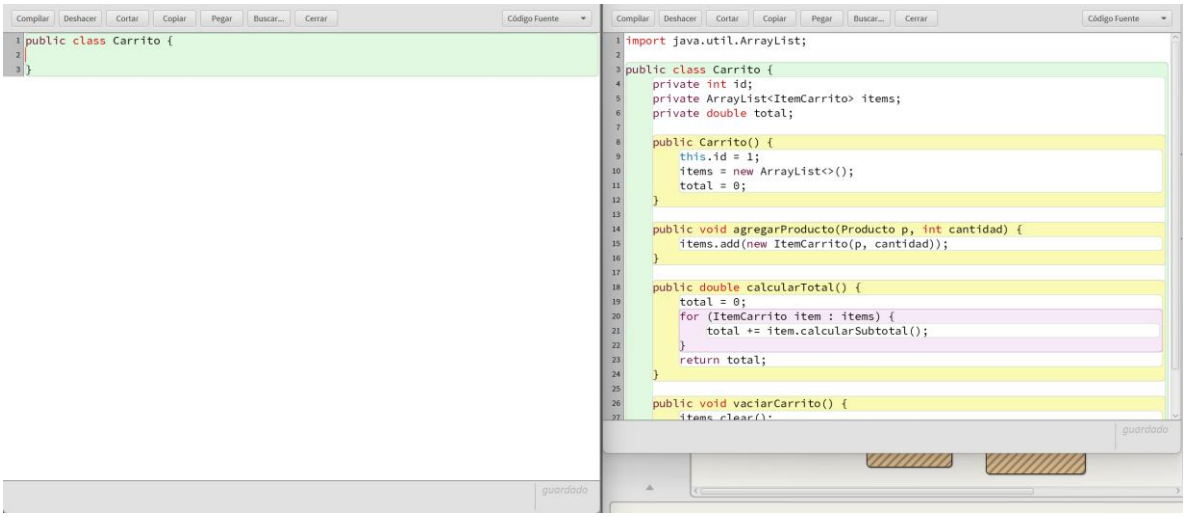
SRP

Cliente extiende a Usuario y añade únicamente información y comportamiento específico del cliente (carrito, dirección, teléfono).

LSP

Cliente puede sustituir a Usuario sin alterar el comportamiento esperado, ya que respeta la estructura base.

Carrito



```java
public class Carrito {

}
```

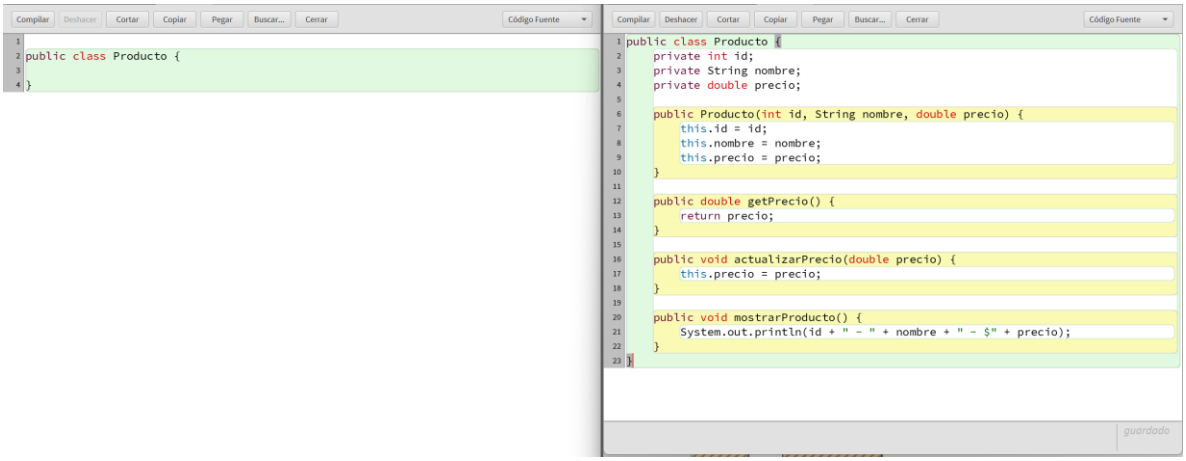
```java
import java.util.ArrayList;

public class Carrito {
    private int id;
    private ArrayList<ItemCarrito> items;
    private double total;

    public Carrito() {
        this.id = 1;
        items = new ArrayList<>();
        total = 0;
    }

    public void agregarProducto(Producto p, int cantidad) {
        items.add(new ItemCarrito(p, cantidad));
    }

    public double calcularTotal() {
        total = 0;
        for (ItemCarrito item : items) {
            total += item.calcularSubtotal();
        }
        return total;
    }

    public void vaciarCarrito() {
        items.clear();
```

SRP

Carrito administra los productos agregados y calcula el total.
No procesa pagos ni genera facturas.

OCP
Puede ampliarse agregando nuevos tipos de ItemCarrito sin modificar su estructura principal.

Producto

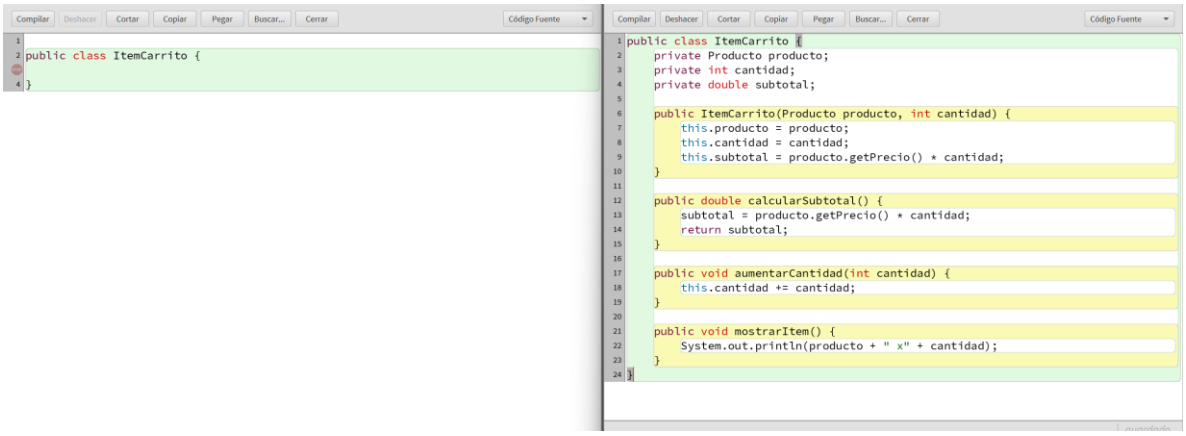


```
public class Producto {

}
```

```
public class Producto {
    private int id;
    private String nombre;
    private double precio;

    public Producto(int id, String nombre, double precio) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
    }

    public double getPrecio() {
        return precio;
    }

    public void actualizarPrecio(double precio) {
        this.precio = precio;
    }

    public void mostrarProducto() {
        System.out.println(id + " - " + nombre + " - $" + precio);
    }
}
```

SRP
Producto solo gestiona información relacionada con un producto (id, nombre, precio) y su modificación.

ItemCarrito



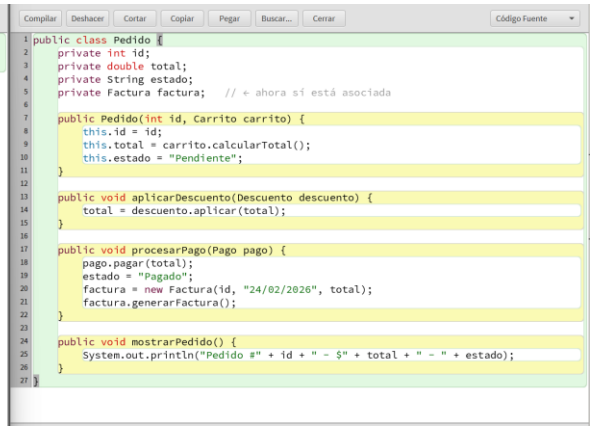
```
public class ItemCarrito {

}
```

```
public class ItemCarrito {
    private Producto producto;
    private int cantidad;
    private double subtotal;

    public ItemCarrito(Producto producto, int cantidad) {
        this.producto = producto;
        this.cantidad = cantidad;
        this.subtotal = producto.getPrecio() * cantidad;
    }

    public double calcularSubtotal() {
        subtotal = producto.getPrecio() * cantidad;
        return subtotal;
    }

    public void aumentarCantidad(int cantidad) {
        this.cantidad += cantidad;
    }

    public void mostrarItem() {
        System.out.println(producto + " x" + cantidad);
    }
}
```

SRP
Se encarga únicamente de representar un producto dentro del carrito junto con su cantidad y subtotal.

Pedido

```
public class Pedido {

}
```



```
public class Pedido {
    private int id;
    private double total;
    private String estado;
    private Factura factura;   // ← ahora sí está asociada

    public Pedido(int id, Carrito carrito) {
        this.id = id;
        this.total = carrito.calcularTotal();
        this.estado = "Pendiente";
    }

    public void aplicarDescuento(Descuento descuento) {
        total = descuento.aplicar(total);
    }

    public void procesarPago(Pago pago) {
        pago.pagar(total);
        estado = "Pagado";
        factura = new Factura(id, "24/02/2026", total);
        factura.generarFactura();
    }

    public void mostrarPedido() {
        System.out.println("Pedido #" + id + " - $" + total + " - " + estado);
    }
}
```

SRP

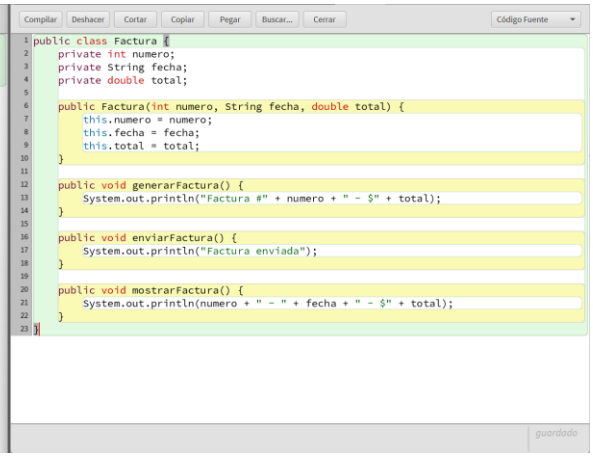
Pedido se encarga exclusivamente de gestionar la compra: calcular total, aplicar descuento y procesar pago.

DIP

Pedido no depende directamente de PagoTarjeta ni PagoEfectivo.
Depende de la abstracción Pago (interfaz).

Factura

```
public class Factura {

}
```



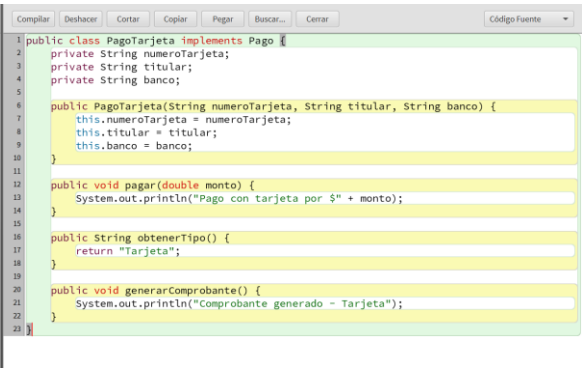
```
public class Factura {
    private int numero;
    private String fecha;
    private double total;

    public Factura(int numero, String fecha, double total) {
        this.numero = numero;
        this.fecha = fecha;
        this.total = total;
    }

    public void generarFactura() {
        System.out.println("Factura #" + numero + " - $" + total);
    }

    public void enviarFactura() {
        System.out.println("Factura enviada");
    }

    public void mostrarFactura() {
        System.out.println(numero + " - " + fecha + " - $" + total);
    }
}
```

Principio aplicado: SRP
Factura se encarga únicamente de generar y mostrar la información de la factura.
No procesa pagos ni aplica descuentos.

Pago

Tarjeta

```
public class PagoTarjeta implements Pago {


}
```



```
public class PagoTarjeta implements Pago {
    private String numeroTarjeta;
    private String titular;
    private String banco;

    public PagoTarjeta(String numeroTarjeta, String titular, String banco) {
        this.numeroTarjeta = numeroTarjeta;
        this.titular = titular;
        this.banco = banco;
    }

    public void pagar(double monto) {
        System.out.println("Pago con tarjeta por $" + monto);
    }

    public String obtenerTipo() {
        return "Tarjeta";
    }

    public void generarComprobante() {
        System.out.println("Comprobante generado - Tarjeta");
    }
}
```

LSP

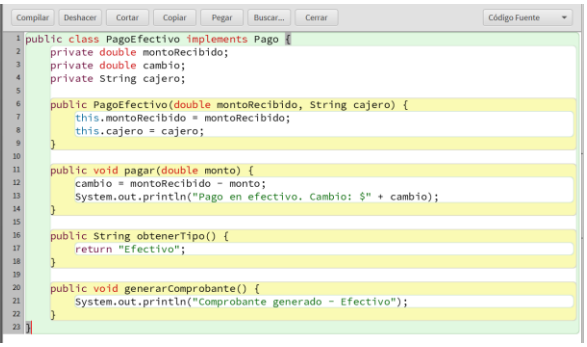
Puede sustituir a Pago sin alterar el funcionamiento del sistema.

DIP

Es una implementación concreta de la abstracción Pago.

Efectivo

```
public class PagoEfectivo implements Pago {

}
```



```
public class PagoEfectivo implements Pago {
    private double montoRecibido;
    private double cambio;
    private String cajero;

    public PagoEfectivo(double montoRecibido, String cajero) {
        this.montoRecibido = montoRecibido;
        this.cajero = cajero;
    }

    public void pagar(double monto) {
        cambio = montoRecibido - monto;
        System.out.println("Pago en efectivo. Cambio: $" + cambio);
    }

    public String obtenerTipo() {
        return "Efectivo";
    }

    public void generarComprobante() {
        System.out.println("Comprobante generado - Efectivo");
    }
}
```

LSP

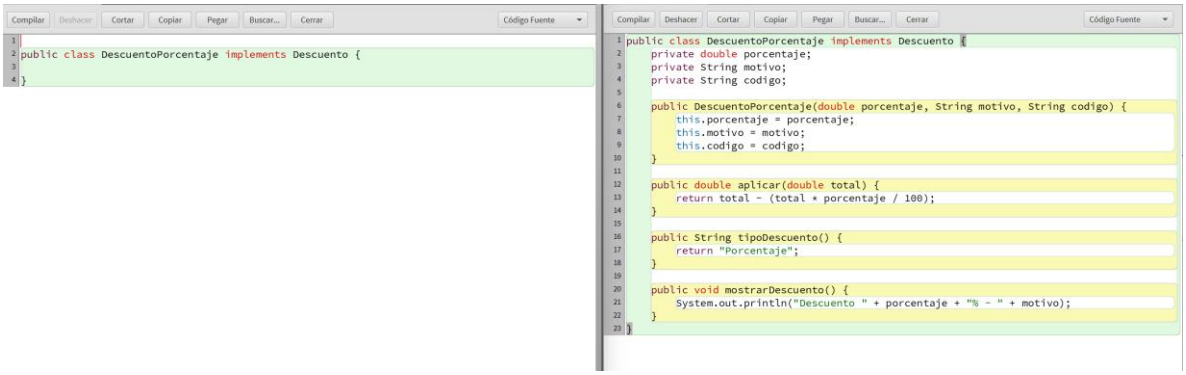
Puede reemplazar a cualquier implementación de Pago.

DIP

Depende de la interfaz Pago, no del sistema directamente.

Descuento

DescuentoPorcentaje



```
public class DescuentoPorcentaje implements Descuento {
}
```

```
public class DescuentoPorcentaje implements Descuento {
    private double porcentaje;
    private String motivo;
    private String codigo;

    public DescuentoPorcentaje(double porcentaje, String motivo, String codigo) {
        this.porcentaje = porcentaje;
        this.motivo = motivo;
        this.codigo = codigo;
    }

    public double aplicar(double total) {
        return total - (total * porcentaje / 100);
    }

    public String tipoDescuento() {
        return "Porcentaje";
    }

    public void mostrarDescuento() {
        System.out.println("Descuento " + porcentaje + "% - " + motivo);
    }
}
```

LSP

Puede sustituir a la interfaz Descuento sin alterar el comportamiento del sistema.

OCP

Se puede crear otro tipo de descuento (por ejemplo, DescuentoFijo) sin modificar el código existente.