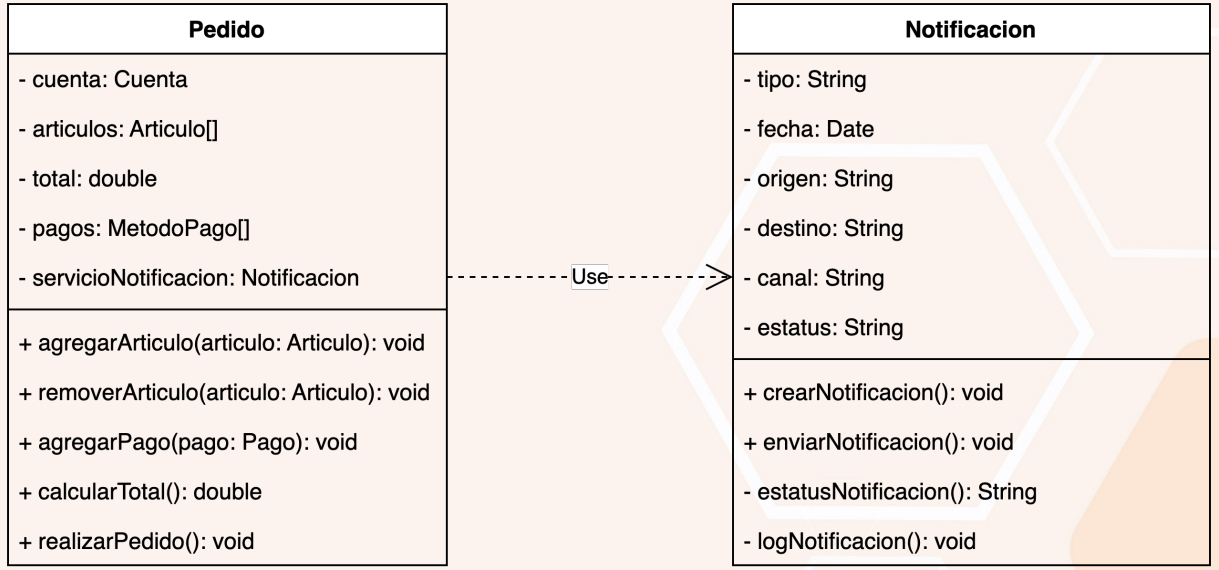
Tipos de relaciones entre clases

Dependencia(“utiliza un”)

Una clase depende de la operacion que ofrece otra clase



Al momento de realizar el pedido, se tiene que notificar al usuario que se ha realizado correctamente.

Tipos de relaciones entre clases

```java
public class Notificacion {
    private String tipo;
    private Date fecha;
    private String origen;
    private String destino;
    private String canal;
    private String estatus;

    public Notificacion(String origen, String destino){
      ....
    }

    public void enviarNotificacion() throws Exception {
        this.estatus="PROCESANDO";
        this.logNotificacion();
        Thread.sleep(2000);
        this.estatus = "ENVIADO";
        this.fecha = new Date();
        this.logNotificacion();
        System.out.println("Notificación enviada a: " + this.destino + " por el canal: " + this.canal);
    }

    private String estatusNotificacion() {
        ...
    }

    private void logNotificacion() {
    }
}
```

```java
public class Pedido {
    private Cuenta cuenta;
    private List<Articulo> articulos;
    private double total;
    private List<MetodoPago> pagos;
    private Notificacion servicioNotificacion;

    public Pedido(Cuenta cuenta) {
        ...
        servicioNotificacion = new Notificacion("PEDIDOSYA.COM", cuenta.obtenerCliente().obtenerCelular());
    }

    public void agregarArticulo(Articulo articulo) {
       ...
    }

    public void removerArticulo(Articulo articulo) {
       ...
    }

    public void agregarPago(MetodoPago pago) {
        ...
    }

    public double calcularTotal() {
        ...
    }

    public void realizarPedido() throws Exception {
            servicioNotificacion.enviarNotificacion();
    }
}
```

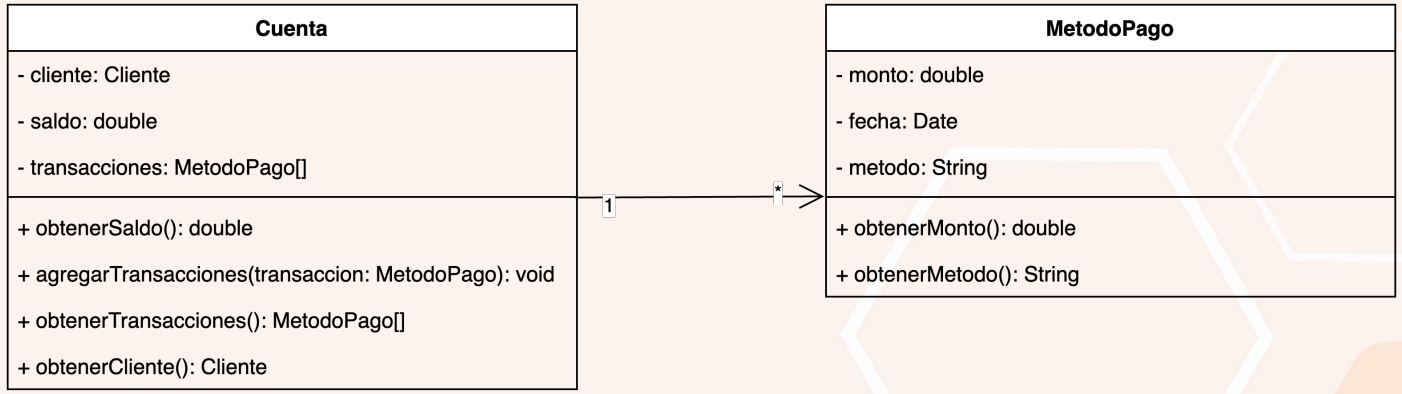
Tipos de relaciones entre clases

Una cuenta tendrá alguna transacción para poder realizar sus operaciones.

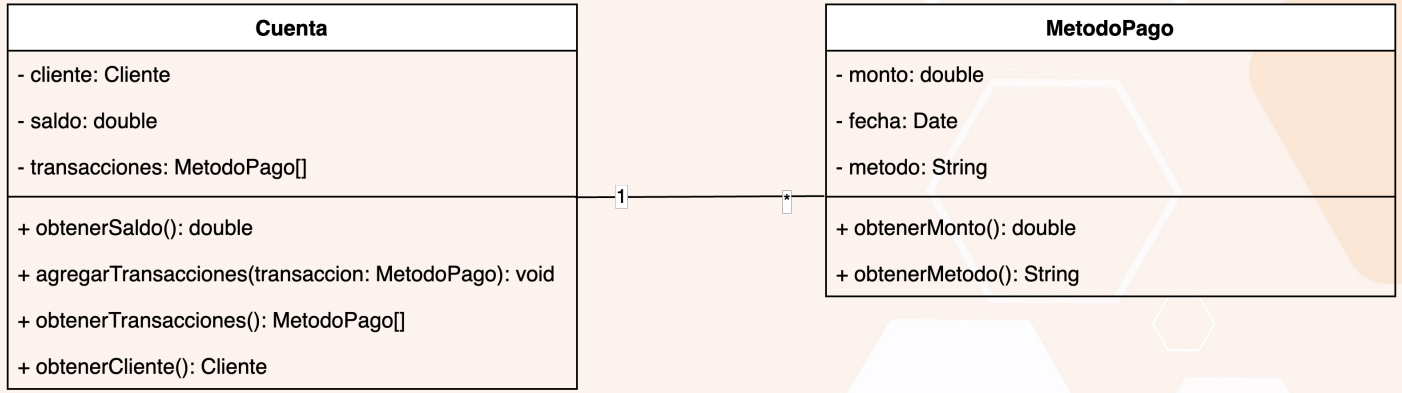
Asociación(“algún”)

Implicita

Explicita



Relacion entre dos unidades independientes.

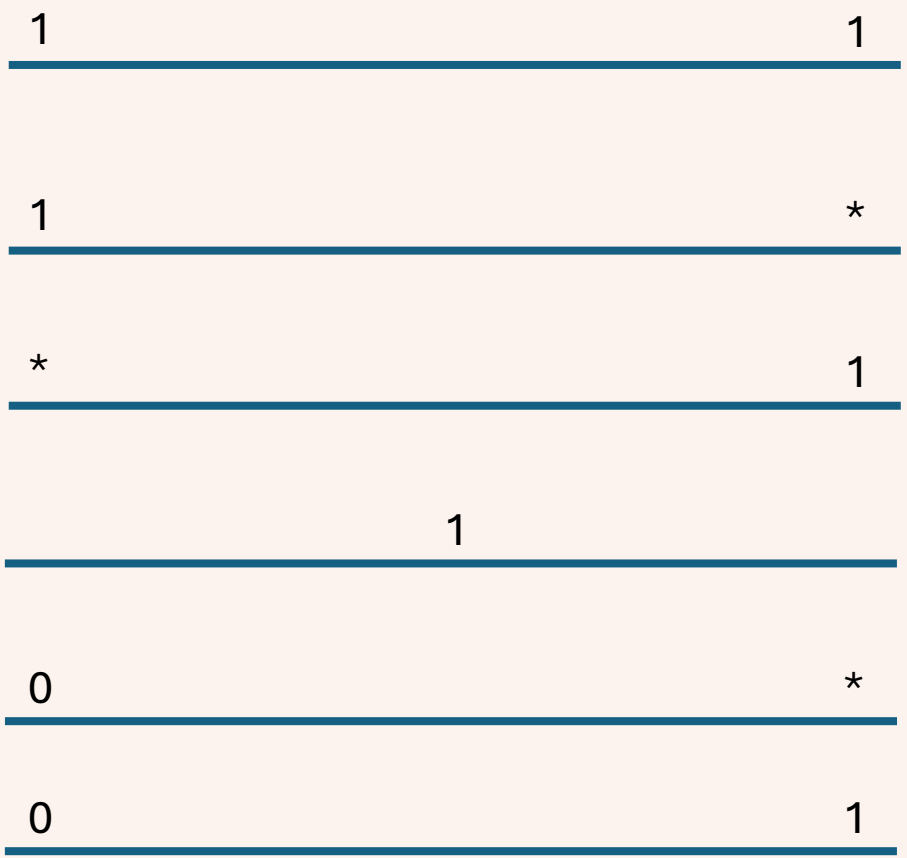


Tipos de relaciones entre clases

Asociación(“algun”)

```java
public class MetodoPago {
    private double monto;
    private Date fecha;
    private String metodo;

    public MetodoPago(double monto, String metodo) {
        this.monto = monto;
        this.fecha = new Date();
        this.metodo = metodo;
    }

    public double obtenerMonto() {
        return monto;
    }

    public String obtenerMetodo() {
        return metodo;
    }
}

public class Cuenta {
    private Cliente cliente;
    private double saldo;
    private List<MetodoPago> transacciones;

    public Cuenta(Cliente cliente){
        ...
    }
    public double obtenerSaldo() {
      ...
    }
    public void agregarTransaccion(MetodoPago transaccion) {
        transacciones.add(transaccion);
        saldo += transaccion.obtenerMonto();
    }
    public List<MetodoPago> obtenerTransacciones() {
        return transacciones;
    }
    public Cliente obtenerCliente() {
       ...
    }
}
```

Multiplicidad



Tipos de relaciones entre clases

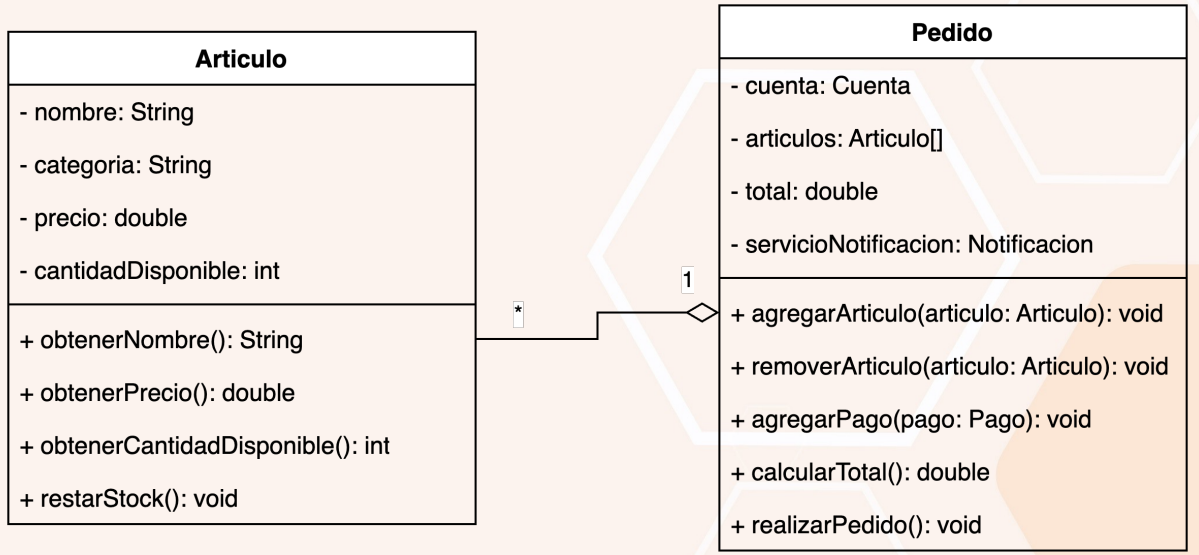
Asociación

Agregación(“tiene un”)

Asociacion independiente entre dos objetos

Si la relacion se rompe, ambos siguen funcionando independientemente.

Un Pedido esta compuesto por multiples Articulos, pero los Articulos pueden existir independientemente del Pedido.



Tipos de relaciones entre clases

Asociación

Agregación(“tiene un”)

```java
public class Articulo {
    private String nombre;
    private String categoria;
    private double precio;
    private int cantidadDisponible;

    public Articulo(String nombre, String categoria, double precio) {
        this.nombre = nombre;
        this.categoria = categoria;
        this.precio = precio;
        this.cantidadDisponible = 100;
    }

    public String obtenerNombre() {
        return nombre;
    }

    public double obtenerPrecio() {
        return precio;
    }

    public int obtenerCantidadDisponible() {
        return cantidadDisponible;
    }

    public void restarStock() {
        cantidadDisponible--;
    }
}

public class Pedido {
    private Cuenta cuenta;
    private List<Articulo> articulos;
    private double total;
    private Notificacion servicioNotificacion;

    public Pedido(Cuenta cuenta) {
        articulos = new ArrayList<Articulo>();
        this.cuenta = cuenta;
        servicioNotificacion = new Notificacion("PEDIDOSYA.COM", cuenta.obtenerCliente().obtenerCelular());
    }

    public void agregarArticulo(Articulo articulo) {
        articulos.add(articulo);
    }

    public void removerArticulo(Articulo articulo) {
        articulos.remove(articulo);
    }

    public void agregarPago(MetodoPago pago) {
        ...
    }

    public double calcularTotal() {
        return articulos.stream().reduce(0.0, (subtotal, articulo) -> subtotal + articulo.obtenerPrecio(), Double::sum);
    }

    public void realizarPedido() throws Exception {
        ...
    }
}
```

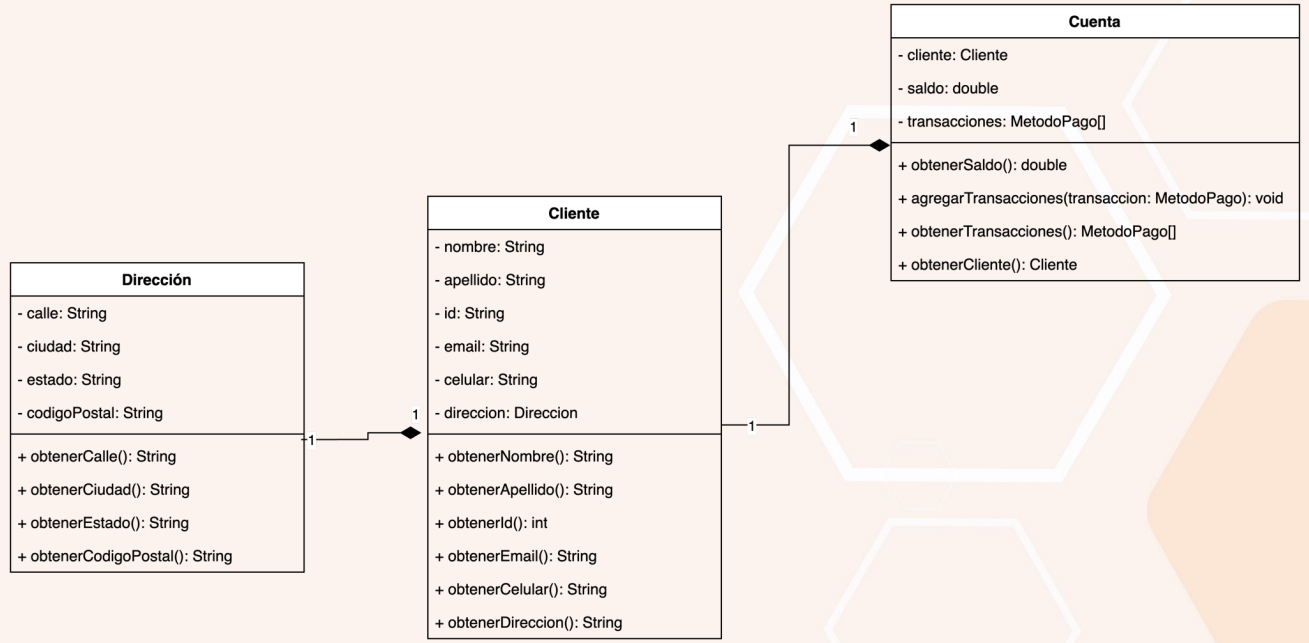
Tipos de relaciones entre clases

Asociación

Composicion(“debe tener un”)

Los objetos asociados no pueden existir independientemente del todo.

Asociacion Fuerte



Tipos de relaciones entre clases

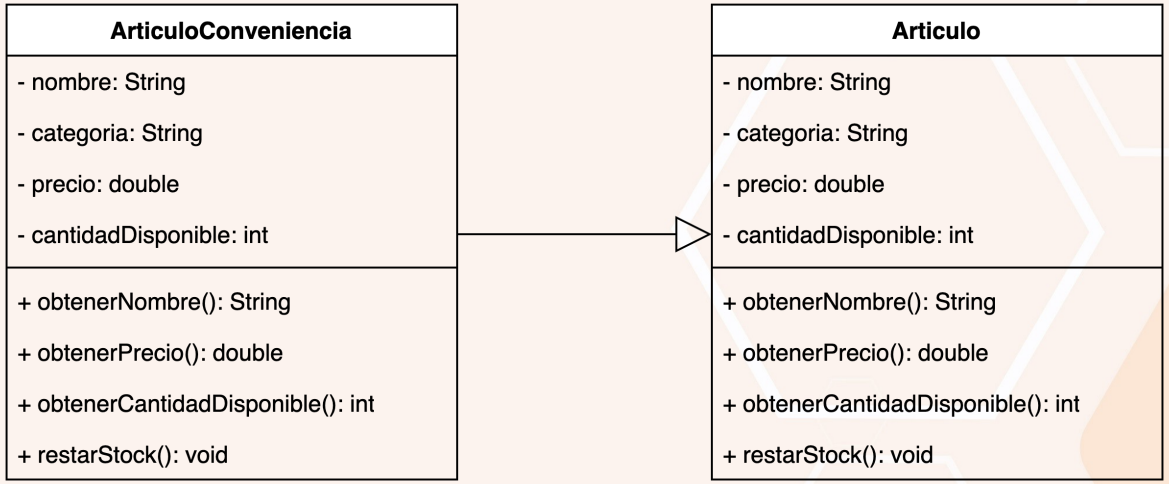
Asociación

Composicion(“se compone de”)

```java
public class Cliente {

    private String nombre;
    private String apellido;
    private String id;
    private String email;
    private String celular;
    private Direccion direccion;

    public Cliente(String nombre, String apellido, String id, String email, String celular, Direccion direccion) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.id = id;
        this.email = email;
        this.celular = celular;
        this.direccion = direccion;
    }

    public String obtenerNombre() {
        return nombre;
    }

    public String obtenerApellido() {
        return apellido;
    }

    public String obtenerId() {
        return id;
    }

    public String obtenerEmail() {
        return email;
    }

    public String obtenerCelular() {
        return celular;
    }

    public Direccion obtenerDireccion() {
        return direccion;
    }


}


public class Direccion {
    private String calle;
    private String ciudad;
    private String estado;
    private String codigoPostal;

    public String obtenerCalle() {
        return calle;
    }

    public String obtenerCiudad() {
        return ciudad;
    }

    public String obtenerEstado() {
        return estado;
    }

    public String obtenerCodigoPostal() {
        return codigoPostal;
    }
}
```

Tipos de relaciones entre clases

Herencia/Generalización(“Es un”)



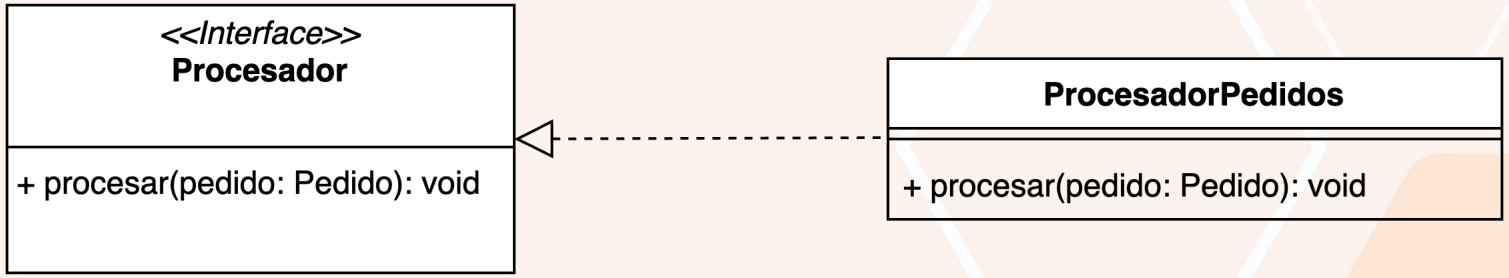
Tipos de relaciones entre clases

Herencia/Generalización(“Es un”)

```java
public class Articulo {
    private String nombre;
    private String categoria;
    private double precio;
    private int cantidadDisponible;

    public Articulo(String nombre, String categoria, double precio) {
        this.nombre = nombre;
        this.categoria = categoria;
        this.precio = precio;
        this.cantidadDisponible = 100;
    }

    ....
}


public class ArticuloConveniencia extends Articulo {

    public ArticuloConveniencia(String nombre, String categoria,  double precio) {
        super(nombre, categoria, precio);
    }

    @Override
    public int obtenerCantidadDisponible() {
        return super.obtenerCantidadDisponible();
    }

    @Override
    public String obtenerNombre() {
        return super.obtenerNombre();
    }

    @Override
    public double obtenerPrecio() {
        return super.obtenerPrecio();
    }

}
```

Tipos de relaciones entre clases

Realización(implementa)



Tipos de relaciones entre clases

Realización(implementa)

```java
public class ProcesadorPedidos implements Procesador {

    @Override
    public void procesar(Pedido pedido) throws Exception {
        pedido.realizarPedido();
    }

}

public interface Procesador {

    public void procesar(Pedido pedido) throws Exception;
}
```