

PROGRAMACIÓN II

Trabajo Práctico 5: Relaciones UML 1 a 1

Alumna: Eugenia Demarchi

DNI: 33346182

Comisión 7

Caso Práctico

Desarrollar los siguientes ejercicios en Java. Cada uno deberá incluir:

- Diagrama UML
- Tipo de relación (asociación, agregación, composición, dependencia)
- Dirección (unidireccional o bidireccional)
- Implementación de las clases con atributos y relaciones definidas

Ejercicios de Relaciones 1 a 1

1. Pasaporte - Foto - Titular

a. Composición: **Pasaporte** → **Foto**

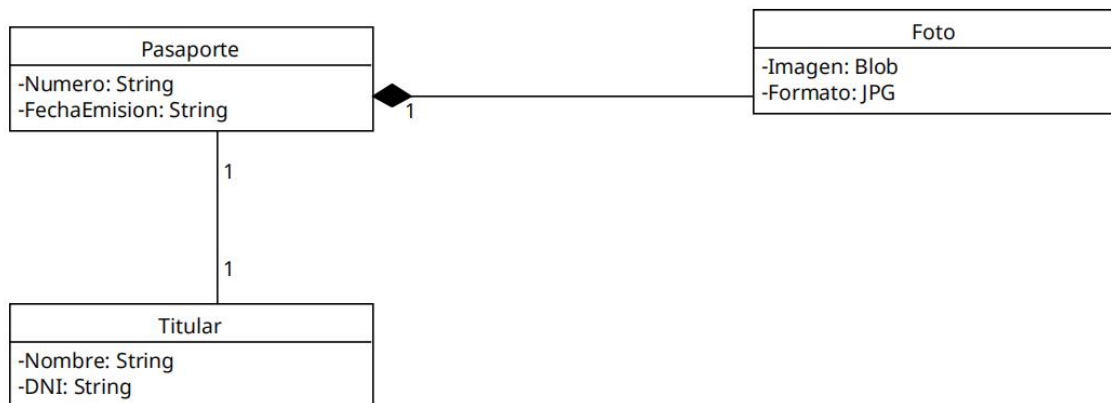
b. Asociación bidireccional: **Pasaporte** ↔ **Titular**

Clases y atributos:

i. Pasaporte: numero, fechaEmision

ii. Foto: imagen, formato

iii. Titular: nombre, dni



a) Composición: **Pasaporte** → **Foto**

```
package Relaciones_1_a_1;
```

```
public class Pasaporte {
    private String numero;
```

```

private String fechaEmision;
private Foto foto;//Atributo de composicion

    public Pasaporte(String numero, String fechaEmision, String imagen, String
formato) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(imagen, formato);
    }

    public Foto getFoto(){
        return foto;
    }

    public void mostrarFoto(){
        System.out.println("El pasaporte " + numero + " tiene fecha de emision: " +
fechaEmision);
        foto.mostrarFoto();
    }
}

```

```

package Relaciones_1_a_1;

```

```

public class Foto {
    private String imagen;
    private String formato;

    public Foto(String imagen, String formato) {
        this.imagen = imagen;
        this.formato = formato;
    }

    public String getImagen() {
        return imagen;
    }

    public String getFormato() {
        return formato;
    }

    public void mostrarFoto(){
        System.out.println("Mostrando foto: " + imagen + " con formato: " +
formato);
    }
}

```

```

package Relaciones_1_a_1;
public class Principal {

    public static void main(String[] args) {
        Pasaporte p1 = new Pasaporte("899090", "29/09/86", "blob", "JPG" );
        p1.mostrarFoto();
    }
}

```

Consola:

El pasaporte 899090 tiene fecha de emision: 29/09/86

Mostrando foto: blob con formato: JPG

b. Asociación bidireccional: **Pasaporte ↔ Titular**

```

package Relaciones_1_a_1;

public class Pasaporte {
    private String numero;
    private String fechaEmision;
    private Foto foto;//Atributo de composicion
    private Titular titular;

    public Pasaporte(String numero, String fechaEmision, String imagen, String formato)
    {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(imagen, formato);
    }

    public Foto getFoto(){
        return foto;
    }

    public void mostrarFoto(){
        System.out.println("El pasaporte " + numero + " tiene fecha de emision: " +
        fechaEmision);
        foto.mostrarFoto();
    }
    //Funcion setter con validacion anti-bucle:
    public void setTitular(Titular titular){
        if (this.titular != titular){
            this.titular = titular;
            if ( titular != null && titular.getPasaporte() != this){
                titular.setPasaporte(this);
            }
        }
    }
}

```

```
public Titular getTitular(){  
    return titular;  
}
```

```
public String getNumero() {  
    return numero;  
}
```

```
public String getFechaEmision() {  
    return fechaEmision;  
}
```

```
public void mostrarPasaporte() {  
    System.out.println("Pasaporte N°: " + numero + " emitido el " + fechaEmision);  
    if (titular != null) {  
        System.out.println("Titular: " + titular.getNombre() + " (DNI: " + titular.getDNI()  
+ ")");  
    }  
    foto.mostrarFoto();  
}
```

```
package Relaciones_1_a_1;
```

```
public class Titular {  
    private String nombre;  
    private String DNI;  
    private Pasaporte pasaporte; // Atributo bidireccional
```

```
    public Titular(String nombre, String DNI) {  
        this.nombre = nombre;  
        this.DNI = DNI;  
    }
```

```
//Funcion setter con validacion anti-bucle:
```

```
public void setPasaporte(Pasaporte pasaporte){  
    if (this.pasaporte != pasaporte){  
        this.pasaporte = pasaporte;  
        if (pasaporte != null && pasaporte.getTitular() != this) {  
            pasaporte.setTitular(this);  
        }  
    }  
}
```

```

}

public Pasaporte getPasaporte(){
    return pasaporte;
}

public String getNombre() {
    return nombre;
}

public String getDNI() {
    return DNI;
}

public void mostrarTitular() {
    System.out.println("Titular: " + nombre + " - DNI: " + DNI);
    if (pasaporte != null) {
        System.out.println("Tiene pasaporte N°: " + pasaporte.getNumero());
    }
}
}

```

```
package Relaciones_1_a_1;
```

```

public class Principal {

    public static void main(String[] args) {
        Pasaporte p1 = new Pasaporte("899090", "29/09/86", "blob", "JPG" );
        Titular perez = new Titular ("Perez", "22787899");

        //ASOCIACION BIDIRECCIONAL: TITULAR----PASAPORTE
        p1.setTitular(perez); //perez.setPasaporte(p1); OPCION PARA CONECTAR LAS CLASES
        //MOSTRAMOS
        p1.mostrarPasaporte();
    }
}

```

CONSOLA:

Pasaporte N^o: 899090 emitido el 29/09/86

Titular: Perez (DNI: 22787899)

Mostrando foto: blob con formato: JPG

2. Celular - Batería - Usuario

a. Agregación: **Celular → Batería**

b. Asociación bidireccional: **Celular ↔ Usuario**

Clases y atributos:

i.

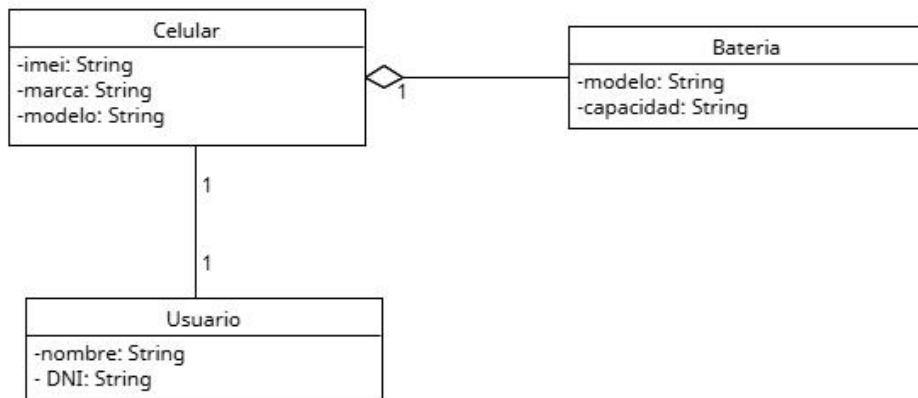
Celular: imei, marca, modelo

ii.

Batería: modelo, capacidad

iii.

Usuario: nombre, dni



a)

```
package relaciones_1_a_1_ejercicio2;
```

```
public class Celular {
```

```
    private String imei;
```

```
    private String marca;
```

```
    private String modelo;
```

```
    private Bateria bateria; //Atributo de agregacion
```

```
    public Celular(String imei, String marca, String modelo) {
        this.imei = imei;
        this.marca = marca;
        this.modelo = modelo;
    }
```

```
//Funcion setter para establecer agregacion:
```

```
    public void setBateria(Bateria bateria){
        this.bateria = bateria;
    }
```

```
//Funcion getter para acceder a la bateria:
```

```
    public Bateria getBateria() {
        return bateria;
    }
```

```

    public String getImei() {
        return imei;
    }

    public String getMarca() {
        return marca;
    }

    public String getModelo() {
        return modelo;
    }
}

```

```

package relaciones_1_a_1_ejercicio2;
public class Bateria {
    private String modelo;
    private String capacidad;

    public Bateria(String modelo, String capacidad) {
        this.modelo = modelo;
        this.capacidad = capacidad;
    }

    public String getModelo() {
        return modelo;
    }

    public String getCapacidad() {
        return capacidad;
    }
}

```

```

package relaciones_1_a_1_ejercicio2;

public class Principal {

    public static void main(String[] args) {
        Bateria bateria = new Bateria("EB-BG991ABY", "4000 mAh");
        Celular celular = new Celular("356789102345678", "Samsung", "Galaxy21");

        //Establecer agregacion:
    }
}

```

```

        celular.setBateria(bateria);
        System.out.println("El celular " + celular.getModelo() + " tiene una bateria " +
bateria.getModelo() + " con capacidad " + bateria.getCapacidad());

    }
}

```

CONSOLA:

run:

El celular Galaxy21 tiene una bateria EB-BG991ABY con capacidad 4000 mAh

BUILD SUCCESSFUL (total time: 0 seconds)

b)

```

package relaciones_1_a_1_ejercicio2;
public class Usuario {
    private String nombre;
    private String DNI;
    private Celular celular;//Atributo bidireccional

```

```

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

```

//Funcion setter con validacion anti-bucle:

```

    public void setCelular(Celular celular) {
        if( this.celular != celular){
            this.celular = celular;
            if (celular != null && celular.getUsuario() != this){
                celular.setUsuario(this);
            }
        }
    }

```

```

    public Celular getCelular(){
        return celular;
    }

```

```

    public String getNombre() {
        return nombre;
    }

```

```

    public String getDNI() {
        return DNI;
    }

```



```
}  
  
}
```

```
package relaciones_1_a_1_ejercicio2;  
public class Celular {  
    private String imei;  
    private String marca;  
    private String modelo;  
    private Bateria bateria; //Atributo de agregacion  
    private Usuario usuario; //Atributo bidireccional  
  
    public Celular(String imei, String marca, String modelo) {  
        this.imei = imei;  
        this.marca = marca;  
        this.modelo = modelo;  
    }  
  
    //Funcion setter para establecer agregacion:  
    public void setBateria(Bateria bateria){  
        this.bateria = bateria;  
    }  
  
    //Funcion getter para acceder a la bateria:  
  
    public Bateria getBateria() {  
        return bateria;  
    }  
  
    public String getImei() {  
        return imei;  
    }  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    //Funcion setter con validacion antibucle:  
    public void setUsuario(Usuario usuario){  
        if (this.usuario != usuario){  
            this.usuario = usuario;  
            if (usuario != null && usuario.getCelular() != this){
```

```

        usuario.setCelular(this);//Establece relacion bidireccional
    }
}

public Usuario getUsuario(){
    return usuario;
}

@Override
public String toString() {
    return "Celular{" + "imei=" + imei + ", marca=" + marca + ", modelo=" + modelo + ",
bateria=" + bateria + ", usuario=" + usuario + '}';
}

}

```

```

package relaciones_1_a_1_ejercicio2;
public class Principal {

    public static void main(String[] args) {
        Bateria bateria = new Bateria("EB-BG991ABY", "4000 mAh");
        Celular celular = new Celular("356789102345678", "Samsung", "Galaxy21");
        Usuario usuario = new Usuario("Eugenia Demarchi", "3389990");

        //Establecer agregacion:
        celular.setBateria(bateria);
        System.out.println("El celular " + celular.getModelo() + " tiene una bateria " +
bateria.getModelo() + " con capacidad " + bateria.getCapacidad());

        //relacion bidireccional: Usar una de las funciones setter (la relacion se establece
automaticamente en ambos sentidos)
        usuario.setCelular(celular);

        //Ver el toString() -no debe ir toString en Usuario tb porque se genera recursion
infinita
        System.out.println(celular);

        // Ver desde el celular quién es su usuario
        System.out.println("Usuario del celular: " + celular.getUsuario().getNombre());

        // Ver desde el usuario cuál es su celular
        System.out.println("Celular del usuario: " + usuario.getCelular().getMarca());

    }
}

```

}

CONSOLA:

run:

El celular Galaxy21 tiene una bateria EB-BG991ABY con capacidad 4000 mAh

Celular{imei=356789102345678, marca=Samsung, modelo=Galaxy21,

bateria=relaciones_1_a_1_ejercicio2.Bateria@54bedef2,

usuario=relaciones_1_a_1_ejercicio2.Usuario@5caf905d}

Usuario del celular: Eugenia Demarchi

Celular del usuario: Samsung

BUILD SUCCESSFUL (total time: 0 seconds)

3. Libro - Autor - Editorial

a. Asociación unidireccional: **Libro → Autor**

b. Agregación: **Libro → Editorial**

Clases y atributos:

i.

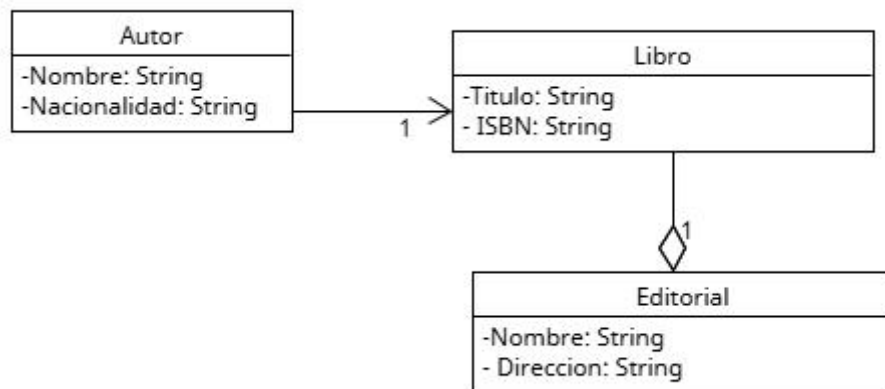
Libro: titulo, isbn

ii.

Autor: nombre, nacionalidad

iii.

Editorial: nombre, dirección



a)

```
ackage Relaciones_1_a_1_ejercicio3;
```

```
public class Libro {
```

```
private String titulo;
```

```
private String isbn;
```

```
private Autor autor;//Atributo de relacion unidireccional 1-1
```

```

public Libro(String titulo, String isbn) {

    this.titulo = titulo;

    this.isbn = isbn;

}

//Funcion setter para establecer la relacion:

public void setAutor(Autor autor) {

    this.autor = autor;

}

// Getter para acceder a datos relacionados

public String getNombreAutor() {

    return autor != null ? autor.getNombre() : "Sin autor";

}

}

```

```

package Relaciones_1_a_1_ejercicio3;

public class Autor {

    private String nombre;

    private String nacionalidad;

    public Autor(String nombre, String nacionalidad) {

        this.nombre = nombre;

        this.nacionalidad = nacionalidad;

    }

    public String getNombre() {

        return nombre;

    }

    public String getNacionalidad() {

        return nacionalidad;

    }

    @Override

    public String toString() {

```

```

        return "Autor{" + "nombre=" + nombre + ", nacionalidad=" + nacionalidad + "}";
    }
}

```

```

package Relaciones_1_a_1_ejercicio3;

public class Principal {

    public static void main(String[] args) {

        Libro libro = new Libro("La condesa sangrienta", "9809");

        Autor autor = new Autor("Alejandra Pizarnik", "argentina");

        //Establecer la relacion unidirecciona:

        libro.setAutor(author);

        // Mostrar por consola el autor con toString()

        System.out.println(author);

        //Mostrar el autor desde libro:

        System.out.println("Autor del libro: " + libro.getNombreAutor());

    }

}

```

CONSOLA:

run:

Autor{nombre=Alejandra Pizarnik, nacionalidad=argentina}

Autor del libro: Alejandra Pizarnik

BUILD SUCCESSFUL (total time: 0 seconds)

B)

```

package Relaciones_1_a_1_ejercicio3;

public class Editorial {

    private String nombre;

    private String direccion;

```

```
private Libro libro; //Atributo de agregacion
```

```
public Editorial(String nombre, String direccion) {  
    this.nombre = nombre;  
    this.direccion = direccion;  
}
```

```
public void setLibro(Libro libro) {  
    this.libro = libro;  
}
```

```
public Libro getLibro() {  
    return libro;  
}
```

```
public String getNombre() {  
    return nombre;  
}  
}
```

```
package Relaciones_1_a_1_ejercicio3;
```

```
public class Libro {
```

```
private String titulo;
```

```
private String isbn;
```

```
private Autor autor; //Atributo de relacion unidireccional 1-1
```

```

public Libro(String titulo, String isbn) {

    this.titulo = titulo;

    this.isbn = isbn;

}

//Funcion setter para establecer la relacion:

public void setAutor(Autor autor) {

    this.autor = autor;

}

// Getter para acceder a datos relacionados

public String getNombreAutor() {

    return autor != null ? autor.getNombre() : "Sin autor";

}

public String getTitulo() {

    return titulo;

}

public String getIsbn() {

    return isbn;

}

}

```

```

package Relaciones_1_a_1_ejercicio3;

```

```

public class Principal {

```

```

    public static void main(String[] args) {

```

```

        Libro libro = new Libro("La condesa sangrienta", "9809");

```

```

        Autor autor = new Autor("Alejandra Pizarnik", "argentina");

```

```

Editorial editorial = new Editorial("Kapeluz", "Aguero 2090");

//Establecer la relacion unidireccional:

libro.setAutor(autor);

// Mostrar por consola el autor con toString()

    System.out.println(autor);

//Mostrar el autor desde libro:

    System.out.println("Autor del libro: " + libro.getNombreAutor());


//Establecer agregacion:

    editorial.setLibro(libro);

    System.out.println("El libro " + libro.getTitulo()+ " de la autora " + libro.getNombreAutor() +
" fue editado por la editorial " + editorial.getNombre());

}

}

```

CONSOLA:

run:

Autor{nombre=Alejandra Pizarnik, nacionalidad=argentina}

Autor del libro: Alejandra Pizarnik

El libro La condesa sangrienta de la autora Alejandra Pizarnik fue editado por la editorial Kapeluz

BUILD SUCCESSFUL (total time: 0 seconds)

4. TarjetaDeCrédito - Cliente - Banco

a. Asociación bidireccional: **TarjetaDeCrédito ↔ Cliente**

b. Agregación: **TarjetaDeCrédito → Banco**

Clases y atributos:

i.

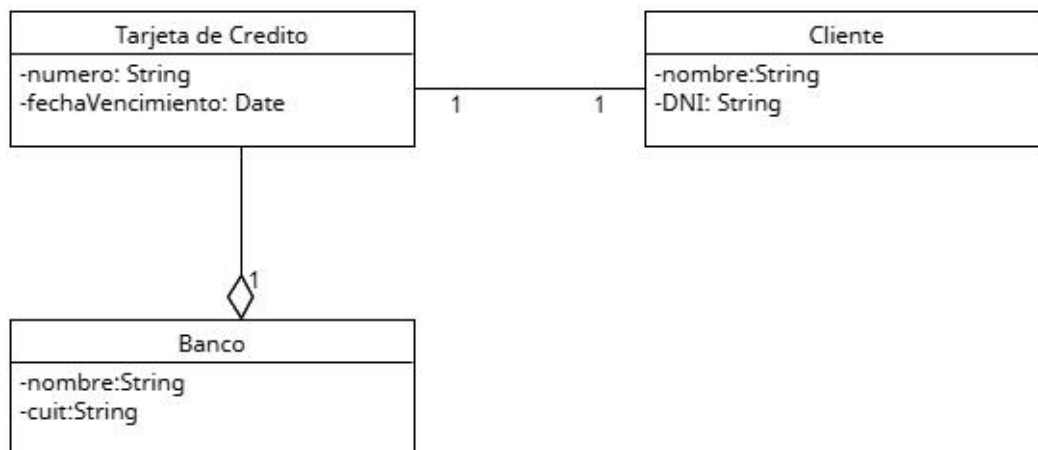
TarjetaDeCrédito: numero, fechaVencimiento

ii.

Cliente: nombre, dni

iii.

Banco: nombre, cuit



A)

```
package relaciones_1_a_1_ejercicio4;
```

```
public class Cliente {
    private String nombre;
    private String DNI;
    private TarjetaDeCredito tarjetaDeCredito;//Bidireccionalidad
```

```
    public Cliente(String nombre, String DNI) {
        this.nombre = nombre;
        this.DNI = DNI;
    }
```

```
//Funcion setter con validacion antibucle:
```

```
    public void setTarjetaDeCredito(TarjetaDeCredito tarjetDeCredito ) {
        if (this.tarjetaDeCredito != tarjetaDeCredito){
            this.tarjetaDeCredito = tarjetDeCredito;
            if (tarjetDeCredito != null && tarjetDeCredito.getCliente() != this){
                tarjetDeCredito.setCliente(this);
            }
        }
    }
```

```
//Funcion getter para acceder a datos relacionados:
```

```
    public TarjetaDeCredito getTarjetaDeCredito(){
        return tarjetaDeCredito;
    }
```

```
    public String getNombre() {
        return nombre;
    }
```

```

    }

    public String getDNI() {
        return DNI;
    }
}

```

```

package relaciones_1_a_1_ejercicio4;

```

```

public class TarjetaDeCredito {
    private String numero;
    private String fechaDeVencimiento;
    private Cliente cliente;
    private Banco banco;// AGREGACION, VER

```

```

    public TarjetaDeCredito(String numero, String fechaDeVencimiento) {
        this.numero = numero;
        this.fechaDeVencimiento = fechaDeVencimiento;
    }

```

```

//Funcion setter con validacion antibucle:

```

```

    public void setCliente(Cliente cliente) {
        if (this.cliente != cliente){
            this.cliente = cliente;
            if (cliente != null && cliente.getTarjetaDeCredito() != this){
                cliente.setTarjetaDeCredito(this);
            }
        }
    }
}

```

```

//Funcion getter para acceder a datos relacionados:

```

```

    public Cliente getCliente(){
        return cliente;
    }

```

```

    public String getNumero() {
        return numero;
    }

```

```

    public String getFechaDeVencimiento() {
        return fechaDeVencimiento;
    }
}

```

```
package relaciones_1_a_1_ejercicio4;
```

```
public class Principal {  
    public static void main(String[] args) {  
        TarjetaDeCredito birza = new TarjetaDeCredito("89999999999", "09/09/35");  
        Cliente c1= new Cliente ("Raul Lozada", "22347767");  
        Banco banco = new Banco("Segovia", " 30-78888898-2");  
  
        //Uso una de las funciones setter para establecer la relacion bidireccional:  
        c1.setTarjetaDeCredito(birza);  
        System.out.println("La tarjeta " + birza.getNumero() + " con vencimiento " +  
        birza.getFechaDeVencimiento() + " es del cliente: " + c1.getNombre());  
    }  
}
```

Consola:

La tarjeta 89999999999 con vencimiento 09/09/35 es del cliente: Raul Lozada
BUILD SUCCESSFUL (total time: 0 seconds)

b. Agregación: [TarjetaDeCrédito](#) → [Banco](#)

```
package relaciones_1_a_1_ejercicio4;
```

```
public class Banco {  
    private String nombreBanco;  
    private String cuit;  
    private TarjetaDeCredito tarjetaDeCredito; //atributo agregacion  
  
    public Banco(String nombre, String cuit) {  
        this.nombreBanco = nombre;  
        this.cuit = cuit;  
    }  
  
    public void setTarjetaDeCredito(TarjetaDeCredito tarjetaDeCredito) {  
        this.tarjetaDeCredito = tarjetaDeCredito;  
    }  
  
    public TarjetaDeCredito getTarjetaDeCredito() {  
        return tarjetaDeCredito;  
    }  
}
```

@Override

```

    public String toString() {
        return "Banco{" + "nombreBanco=" + nombreBanco + ", cuit=" + cuit + ",
tarjetaDeCredito=" + tarjetaDeCredito + '}';
    }

    public String getNombreBanco() {
        return nombreBanco;
    }

    public String getCuit() {
        return cuit;
    }
}

```

```

package relaciones_1_a_1_ejercicio4;

```

```

public class TarjetaDeCredito {
    private String numero;
    private String fechaDeVencimiento;
    private Cliente cliente;

    public TarjetaDeCredito(String numero, String fechaDeVencimiento) {
        this.numero = numero;
        this.fechaDeVencimiento = fechaDeVencimiento;
    }
}

```

```

//Funcion setter con validacion antibucle:

```

```

    public void setCliente(Cliente cliente) {
        if (this.cliente != cliente){
            this.cliente = cliente;
            if (cliente != null && cliente.getTarjetaDeCredito() != this){
                cliente.setTarjetaDeCredito(this);
            }
        }
    }
}

```

```

//Funcion getter para acceder a datos relacionados:

```

```

    public Cliente getCliente(){
        return cliente;
    }
}

```

```

    public String getNumero() {
        return numero;
    }
}

```

```

    public String getFechaDeVencimiento() {
        return fechaDeVencimiento;
    }
}

```

```

package relaciones_1_a_1_ejercicio4;

```

```

public class Principal {
    public static void main(String[] args) {
        TarjetaDeCredito birza = new TarjetaDeCredito("89999999999", "09/09/35");
        Cliente c1= new Cliente ("Raul Lozada", "22347767");
        Banco banco = new Banco("Segovia", " 30-78888898-2");

        //Uso una de las funciones setter para establecer la relacion bidireccional:
        c1.setTarjetaDeCredito(birza);
        System.out.println("La tarjeta " + birza.getNumero() + " con vencimiento " +
            birza.getFechaDeVencimiento() + " es del cliente: " + c1.getNombre());

        //Agregacion
        banco.setTarjetaDeCredito(birza);
        System.out.println("La tarjeta " + birza.getNumero() + " pertenece al banco " +
            banco.getNombreBanco());
    }
}

```

Consola:

run:

La tarjeta 89999999999 con vencimiento 09/09/35 es del cliente: Raul Lozada

La tarjeta 89999999999 pertenece al banco Segovia

BUILD SUCCESSFUL (total time: 0 seconds)

5. Computadora - PlacaMadre - Propietario

a. Composición: **Computadora → PlacaMadre**

b. Asociación bidireccional: **Computadora ↔ Propietario**

Clases y atributos:

i.

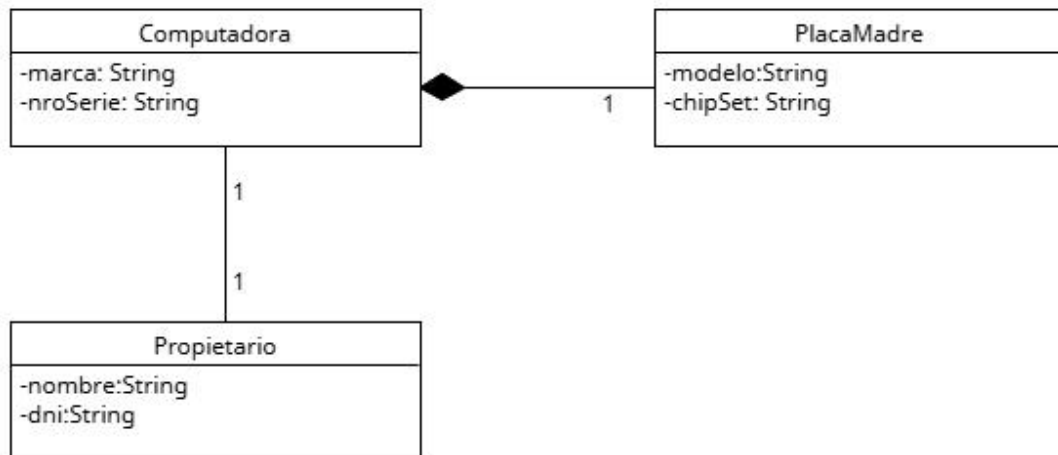
Computadora: marca, numeroSerie

ii.

PlacaMadre: modelo, chipset

iii.

Propietario: nombre, dni



a. Composición: **Computadora** → **PlacaMadre**

```

package relaciones_1_a_1_ejercicio5;
public class Computadora {
    private String marca;
    private String nroSerie;
    private PlacaMadre placaMadre;//atributo de composicion

    public Computadora(String marca, String nroSerie, String modelo, String chipSet) {
        this.marca = marca;
        this.nroSerie = nroSerie;
        //Creacion directa de la parte (composicion)
        this.placaMadre = new PlacaMadre(modelo, chipSet);
    }

    public PlacaMadre getPlacaMadre(){
        return placaMadre;
    }

    public void mostrarInformacion(){
        System.out.println("Marca: " + marca);
        placaMadre.mostrar();
    }

    @Override
    public String toString() {
        return "Computadora{" + "marca=" + marca + ", nroSerie=" + nroSerie + ",
        placaMadre=" + placaMadre + '}';
    }
}
  
```

```
}
```

```
package relaciones_1_a_1_ejercicio5;
```

```
public class PlacaMadre {  
    private String modelo;  
    private String chipSet;  
  
    public PlacaMadre(String modelo, String chipSet) {  
        this.modelo = modelo;  
        this.chipSet = chipSet;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public String getChipSet() {  
        return chipSet;  
    }  
  
    public void mostrar(){  
        System.out.println("Mostrando modelo de placa madre: " + modelo);  
    }  
  
    @Override  
    public String toString() {  
        return "PlacaMadre{" + "modelo=" + modelo + ", chipSet=" + chipSet + '}';  
    }  
}
```

```
package relaciones_1_a_1_ejercicio5;
```

```
public class Principal {  
    public static void main(String[] args) {  
        Computadora computadora = new Computadora("Dell OptiPlex 7090",  
            "CN0A1B2C3D4E567FG", "ASUS PRIME B560M-A", "Intel B560");  
        System.out.println(computadora);  
        computadora.mostrarInformacion();  
    }  
}
```

Consola:

run:

```
Computadora{marca=Dell OptiPlex 7090, nroSerie=CN0A1B2C3D4E567FG,
placaMadre=PlacaMadre{modelo=ASUS PRIME B560M-A, chipSet=Intel B560}}
Marca: Dell OptiPlex 7090
Mostrando modelo de placa madre: ASUS PRIME B560M-A
BUILD SUCCESSFUL (total time: 0 seconds)
```

b. Asociación bidireccional: **Computadora ↔ Propietario**

```
package relaciones_1_a_1_ejercicio5;
```

```
public class Computadora {
    private String marca;
    private String nroSerie;
    private PlacaMadre placaMadre;//atributo de composicion
    private Propietario propietario;//atributo bidireccional

    public Computadora(String marca, String nroSerie, String modelo, String chipSet) {
        this.marca = marca;
        this.nroSerie = nroSerie;
        //Creacion directa de la parte (composicion)
        this.placaMadre = new PlacaMadre(modelo, chipSet);
    }

    public PlacaMadre getPlacaMadre(){
        return placaMadre;
    }

    public void mostrarInformacion(){
        System.out.println("Marca: " + marca);
        placaMadre.mostrar();
    }

    @Override
    public String toString() {
        return "Computadora{" + "marca=" + marca + ", nroSerie=" + nroSerie + ", placaMadre=" +
        placaMadre + '}';
    }

    public String getMarca() {
        return marca;
    }

    //Funcion setter con validacion anti-bucle
    public void setPropietario(Propietario propietario){
        if (this.propietario != propietario){
            this.propietario =propietario;
            if (propietario !=null && propietario.getComputadora() != this){
                propietario.setComputadora(this);
            }
        }
    }
}
```



```

    }
}
}
public Propietario getPropietario(){
    return propietario;
}

public String getNombrePropietario(){
    return propietario !=null ? propietario.getNombre() : "Sin propietario";
}
}

```

```

package relaciones_1_a_1_ejercicio5;
public class Propietario {
    private String nombre;
    private String DNI;
    private Computadora computadora;//ATRIBUTO BIDIRECCIONAL

    public Propietario(String nombre, String DNI) {
        this.nombre = nombre;
        this.DNI = DNI;
    }

    public void setComputadora(Computadora computadora){
        if (this.computadora != computadora){
            this.computadora = computadora;
            if (computadora != null && computadora.getPropietario() !=this){
                computadora.setPropietario(this);//Establecer relacion bidireccional
            }
        }
    }

    //Funcion getter para acceder a los datos relacionados:
    public Computadora getComputadora(){
        return computadora;
    }

    public String getMarcaComputadora(){
        return computadora !=null ? computadora.getMarca() : "Sin marca";
    }

    public String getNombre() {
        return nombre;
    }

    public String getDNI() {
        return DNI;
    }

    @Override

```

```

    public String toString() {
        return "Propietario{" + "nombre=" + nombre + ", DNI=" + DNI + ", computadora=" +
computadora + '}';
    }
}

```

```

package relaciones_1_a_1_ejercicio5;
public class Principal {
    public static void main(String[] args) {
        Computadora computadora = new Computadora("Dell OptiPlex 7090",
"CN0A1B2C3D4E567FG", "ASUS PRIME B560M-A", "Intel B560");
        Propietario p1 = new Propietario("Tomas Perez", "44567888");

        System.out.println(computadora);
        computadora.mostrarInformacion();

        //usar UNA de las funciones setter para crear la bidireccionalidad:
        p1.setComputadora(computadora);
        System.out.println("Desde Propietario: " + p1.getNombre() +
            " tiene la computadora " + p1.getMarcaComputadora());

        System.out.println("Desde Computadora: " + computadora.getMarca() +
            " pertenece a " + computadora.getNombrePropietario());
    }
}

```

Consola:

run:

```

Computadora{marca=Dell OptiPlex 7090, nroSerie=CN0A1B2C3D4E567FG,
placaMadre=PlacaMadre{modelo=ASUS PRIME B560M-A, chipSet=Intel B560}}
Marca: Dell OptiPlex 7090
Mostrando modelo de placa madre: ASUS PRIME B560M-A
Desde Propietario: Tomas Perez tiene la computadora Dell OptiPlex 7090
Desde Computadora: Dell OptiPlex 7090 pertenece a Tomas Perez
BUILD SUCCESSFUL (total time: 0 seconds)

```

6. Reserva - Cliente - Mesa

a. Asociación unidireccional: [Reserva → Cliente](#)

b. Agregación: [Reserva → Mesa](#)

Clases y atributos:

i.

Reserva: fecha, hora

ii.

Cliente: nombre, telefono

iii.

Mesa: numero, capacidad

a. Asociación unidireccional: [Reserva → Cliente](#)

```

package relaciones_1_a_1_ejercicio6;
public class Reserva {
    private String fecha;
    private String hora;
    private Cliente cliente;//Atributo de relacion

    public Reserva(String fecha, String hora) {
        this.fecha = fecha;
        this.hora = hora;
    }
    //Funcion setter para establecer relacion
    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    //Funcion getter para acceder a datos relacionados
    public String getNombreCliente(){
        return cliente !=null ? cliente.getNombre() : "Sin cliente";
    }

    @Override
    public String toString() {
        return "Reserva{" + "fecha=" + fecha + ", hora=" + hora + ", cliente=" + cliente + '}';
    }

    public String getFecha() {
        return fecha;
    }

    public String getHora() {
        return hora;
    }
}

```

```

package relaciones_1_a_1_ejercicio6;
public class Cliente {//clase independiente
    private String nombre;
    private String telefono;

    public Cliente(String nombre, String telefono) {
        this.nombre = nombre;
        this.telefono = telefono;
    }
}

```

```

    public String getNombre() {
        return nombre;
    }

    public String getTelefono() {
        return telefono;
    }
}

package relaciones_1_a_1_ejercicio6;

public class Principal {
    public static void main(String[] args) {
        Cliente c1 = new Cliente ("Pedro Jota", "1180909090");
        Reserva r1 = new Reserva("11/10/25", "21:00");

        //Establecer la relacion unidireccional
        r1.setCliente(c1);
        System.out.println("La reserva con fecha " + r1.getFecha() + " para el cliente: " +
r1.getNombreCliente() + " es a las " + r1.getHora() + " hs.");
    }
}

```

b. Agregación: **Reserva** → **Mesa**

```

package relaciones_1_a_1_ejercicio6;

public class Reserva { //clase contenedora de la agregacion
    private String fecha;
    private String hora;
    private Cliente cliente; //Atributo de relacion
    private Mesa mesa; //atributo de la agregacion

    public Reserva(String fecha, String hora) {
        this.fecha = fecha;
        this.hora = hora;
    }
    //Funcion setter para establecer relacion
    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }
}

```

```

//Funcion getter para acceder a datos relacionados
public String getNombreCliente(){
    return cliente !=null ? cliente.getNombre() : "Sin cliente";
}

@Override
public String toString() {
    return "Reserva{" + "fecha=" + fecha + ", hora=" + hora + ", cliente=" + cliente + '}';
}

public String getFecha() {
    return fecha;
}

public String getHora() {
    return hora;
}

//Funcion setter para establecer la agregacion:
public void setMesa(Mesa mesa){
    this.mesa = mesa;
}

public Mesa getMesa(){
    return mesa;
}
}

```

```

package relaciones_1_a_1_ejercicio6;

```

```

public class Mesa { //clase contenida de la Agregacion
    private String numero;
    private String capacidad;

```

```

    public Mesa(String numero, String capacidad) {
        this.numero = numero;
        this.capacidad = capacidad;
    }

```

```

    public String getNumero() {
        return numero;
    }

```

```

    public String getCapacidad() {

```

```

        return capacidad;
    }
}

```

```

package relaciones_1_a_1_ejercicio6;

```

```

public class Principal {
    public static void main(String[] args) {
        Cliente c1 = new Cliente ("Pedro Jota", "1180909090");
        Reserva r1 = new Reserva("11/10/25", "21:00");
        Mesa mesa1 = new Mesa("20", "6 personas");

        //Establecer la relacion unidireccional
        r1.setCliente(c1);
        System.out.println("La reserva con fecha " + r1.getFecha() + " para el cliente: " +
r1.getNombreCliente() + " es a las " + r1.getHora() + " hs.");

        //Establecer la agregacion:
        r1.setMesa(mesa1);
        System.out.println("La reserva para " + r1.getNombreCliente() + " sera en la mesa "
+ mesa1.getNumero() + " con capacidad para " + mesa1.getCapacidad());
    }
}

```

7. Vehículo - Motor - Conductor

a. Agregación: **Vehículo → Motor**

b. Asociación bidireccional: **Vehículo ↔ Conductor**

Clases y atributos:

i.

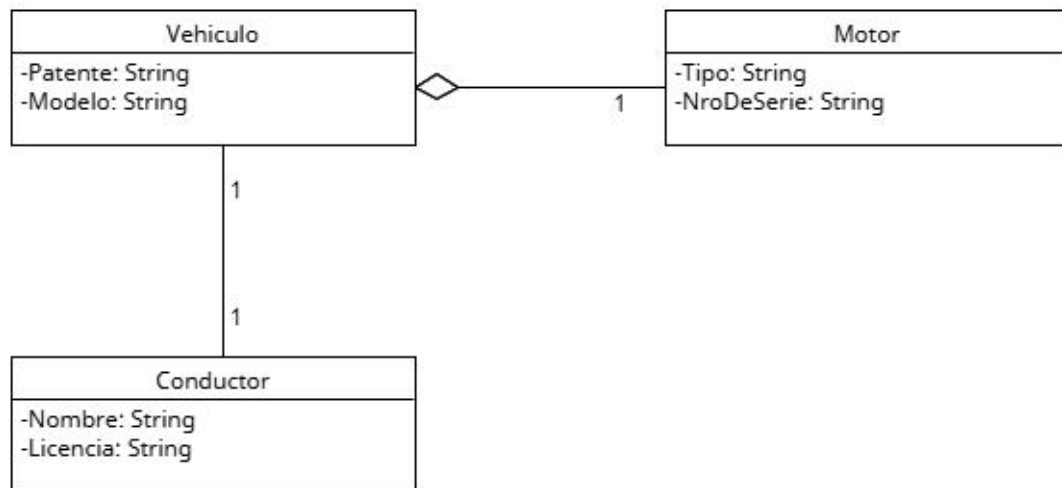
Vehículo: patente, modelo

ii.

Motor: tipo, numeroSerie

iii.

Conductor: nombre, licencia



a. Agregación: **Vehículo → Motor**

```

package relaciones_1_a_1_ejercicio7;
public class Vehiculo {
    private String patente;
    private String modelo;
    private Motor motor;// atributo de agregacion

    public Vehiculo(String patente, String modelo) {
        this.patente = patente;
        this.modelo = modelo;
    }

    //Funcion setter para establecer la agregacion:

    public void setMotor(Motor motor) {
        this.motor = motor;
    }

    //Funcion getter para acceder al motor:
    public Motor getMotor(){
        return motor;
    }

    public void encender(){
        if (motor != null){
            motor.arrancar();
        } else{
            System.out.println("Auto sin motor");
        }
    }
}
  
```

```

    public String getPatente() {
        return patente;
    }

    public String getModelo() {
        return modelo;
    }
    //encapsulamiento de atributos de motor:
    public String getTipoMotor(){
        if (motor !=null){
            return motor.getTipo();
        } else {
            return "Sin motor";
        }
    }

    public String getNroSerieMotor(){
        if (motor != null){
            return motor.getNroDeSerie();
        } else {
            return "Sin numero de serie";
        }
    }
}

```

```

package relaciones_1_a_1_ejercicio7;

```

```

public class Motor {
    private String tipo;
    private String nroDeSerie;

    public Motor(String tipo, String nroDeSerie) {
        this.tipo = tipo;
        this.nroDeSerie = nroDeSerie;
    }

    public String getTipo() {
        return tipo;
    }

    public String getNroDeSerie() {
        return nroDeSerie;
    }
}

```



```

public void arrancar(){
    System.out.println("Motor arrancado");
}
}

```

```

package relaciones_1_a_1_ejercicio7;

```

```

public class Principal {
    public static void main(String[] args) {
        Motor motor = new Motor("V6", "3000");
        Vehiculo v1 = new Vehiculo("IFJ303", "Civic");

        //Establecer la aghregacion
        v1.setMotor(motor);
        System.out.println("El vehiculo con patente " + v1.getPatente() + " tiene un motor
del tipo: " + v1.getTipoMotor());
    }
}

```

Consola:

run:

El vehiculo con patente IFJ303 tiene un motor del tipo: V6
BUILD SUCCESSFUL (total time: 0 seconds)

b. Asociación bidireccional: **Vehículo ↔ Conductor**

```

package relaciones_1_a_1_ejercicio7;

```

```

public class Vehiculo {
    private String patente;
    private String modelo;
    private Motor motor;// atributo de agregacion
    private Conductor conductor; // atributo bidireccional

    public Vehiculo(String patente, String modelo) {
        this.patente = patente;
        this.modelo = modelo;
    }
}

```

//Funcion setter para establecer la agregacion:

```

    public void setMotor(Motor motor) {

```

```

        this.motor = motor;
    }

    //Funcion getter para acceder al motor:
    public Motor getMotor(){
        return motor;
    }

    public void encender(){
        if (motor != null){
            motor.arrancar();
        } else{
            System.out.println("Auto sin motor");
        }
    }

    public String getPatente() {
        return patente;
    }

    public String getModelo() {
        return modelo;
    }

    //encapsulamiento de atributos de motor:
    public String getTipoMotor(){
        if (motor !=null){
            return motor.getTipo();
        } else {
            return "Sin motor";
        }
    }

    public String getNroSerieMotor(){
        if (motor != null){
            return motor.getNroDeSerie();
        } else {
            return "Sin numero de serie";
        }
    }

    //Funcion setter con validacion antibucle
    public void setConductor(Conductor conductor){
        if (this.conductor != conductor){
            this.conductor = conductor;
            if (conductor != null && conductor.getVehiculo() != this){
                conductor.setVehiculo(this);
            }
        }
    }

```

```

    }
}
public Conductor getConductor(){
    return conductor;
}
public String getNombreConductor(){
    return conductor != null ? conductor.getNombre() : "Sin conductor";
}
}

```

```

package relaciones_1_a_1_ejercicio7;
public class Conductor {
    private String nombre;
    private String licencia;
    private Vehiculo vehiculo;//Atributo bidireccional

    public Conductor(String nombre, String licencia) {
        this.nombre = nombre;
        this.licencia = licencia;
    }

    //Funcion setter con validacion antibucle:
    public void setVehiculo(Vehiculo vehiculo){
        if (this.vehiculo != vehiculo){
            this.vehiculo = vehiculo;
        }
        if (vehiculo != null && vehiculo.getConductor() != this){
            vehiculo.setConductor(this);//Establece relacion bidireccional
        }
    }

    //Funcion getter para acceder a datos relacionados:
    public Vehiculo getVehiculo(){
        return vehiculo;
    }

    public String getPatenteVehiculo(){
        return vehiculo != null ? vehiculo.getPatente() : "Sin vehiculo";
    }

    public String getNombre() {
        return nombre;
    }
}

```

```
}
```

```
package relaciones_1_a_1_ejercicio7;
public class Principal {
    public static void main(String[] args) {
        Motor motor = new Motor("V6", "3000");
        Vehiculo v1 = new Vehiculo("IFJ303", "Civic");
        Conductor c1 = new Conductor("Laura Juarez", "jjjik9090");

        //Establecer la aghregacion
        v1.setMotor(motor);
        System.out.println("El vehiculo con patente " + v1.getPatente() + " tiene un motor
del tipo: " + v1.getTipoMotor());
        //relacion bidireccional conductor-vehiculo
        c1.setVehiculo(v1);
        System.out.println("La conductora " + v1.getNombreConductor() + " esta manejando
un Honda " + v1.getModelo() + " con patente " + v1.getPatente());
    }
}
```

Consola:

run:

El vehiculo con patente IFJ303 tiene un motor del tipo: V6

La conductora Laura Juarez esta manejando un Honda Civic con patente IFJ303

BUILD SUCCESSFUL (total time: 0 seconds)

8. Documento - FirmaDigital - Usuario

a. Composición: [Documento](#) → [FirmaDigital](#)

b. Agregación: [FirmaDigital](#) → [Usuario](#)

Clases y atributos:

i.

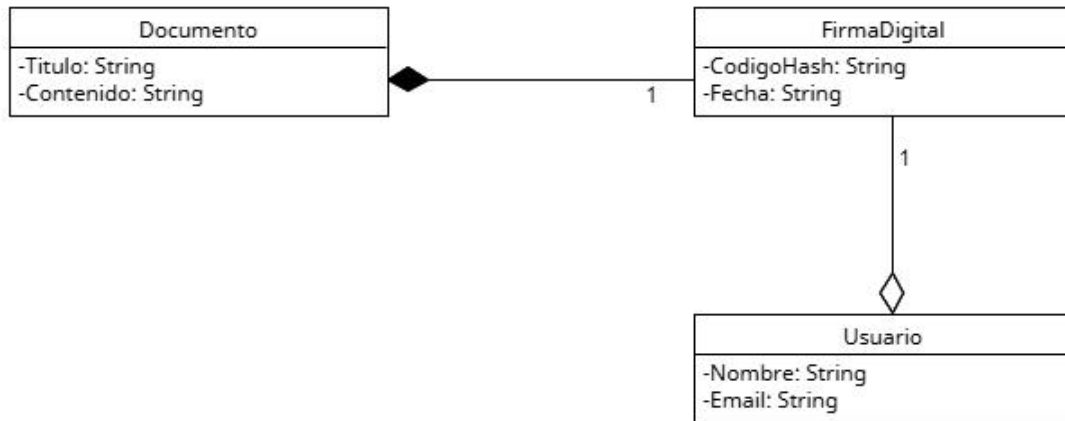
Documento: titulo, contenido

ii.

FirmaDigital: codigoHash, fecha

iii.

Usuario: nombre, email



a. Composición: **Documento → FirmaDigital**

```

package relaciones_1_a_1_ejercicio8;
public class Documento { //Clase contenedora
    private String titulo;
    private String contenido;
    private FirmaDigital firmaDigital; //Atributo de composicion

    public Documento(String titulo, String Contenido, String codigoHash, String fecha) {
        this.titulo = titulo;
        this.contenido = Contenido;
        //creacion directa de la parte:
        this.firmaDigital = new FirmaDigital(codigoHash, fecha);
    }

    //Funcion getter para acceder a la parte:
    public FirmaDigital getFirmaDigital(){
        return firmaDigital;
    }

    //Funcion que usa la parte
    public void mostrarInformacion(){
        System.out.println("Titulo: " + titulo);
        firmaDigital.mostrar();
    }
}
  
```

```

package relaciones_1_a_1_ejercicio8;
public class FirmaDigital { //clase contenida de la composicion
    private String codigoHash;
    private String fecha;

    public FirmaDigital(String codigoHash, String fecha) {
        this.codigoHash = codigoHash;
        this.fecha = fecha;
    }
}
  
```

```

    }

    public String getCodigoHash() {
        return codigoHash;
    }

    public String getFecha() {
        return fecha;
    }

    public void mostrar(){
        System.out.println("Mostrando firma digital: " + codigoHash);
    }
}

package relaciones_1_a_1_ejercicio8;
public class Principal {
    public static void main(String[] args) {
        Documento doc1 = new Documento ("Contrato de locacion", "alquiler propiedad",
"#456hhy", "12/12/25");
        doc1.mostrarInformacion();
    }
}

```

Consola:

run:

Titulo: Contrato de locacion

Mostrando firma digital: #456hhy

BUILD SUCCESSFUL (total time: 0 seconds)

b. Agregación: **FirmaDigital** → **Usuario**

```

package relaciones_1_a_1_ejercicio8;

public class Usuario {
    private String nombre;
    private String email;
    private FirmaDigital firmaDigital;

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

```

```

    }

    public String getNombre() {
        return nombre;
    }

    public String getEmail() {
        return email;
    }
    //FUNCION SETTER PARA ESTABLECER LA AGREGACION:
    public void setFirmaDigital(FirmaDigital firmaDigital){
        this.firmaDigital = firmaDigital;
    }

    public FirmaDigital getFirmaDigital(){
        return firmaDigital;
    }

    //Funcion que usa la clase agregada:
    public String acceder() {
        if (firmaDigital != null) {
            return "Hash: " + firmaDigital.getCodigoHash() + ", Fecha: " +
firmaDigital.getFecha();
        } else {
            return "Documento sin firma";
        }
    }
}

```

```

package relaciones_1_a_1_ejercicio8;
public class FirmaDigital { //clase contenida de la composicion y de la agregacion
    private String codigoHash;
    private String fecha;

```

```

        public FirmaDigital(String codigoHash, String fecha) {
            this.codigoHash = codigoHash;
            this.fecha = fecha;
        }

```

```

        public String getCodigoHash() {
            return codigoHash;
        }

```

```

        public String getFecha() {
            return fecha;
        }

```

```

    }

    public void mostrar(){
        System.out.println("Mostrando firma digital: " + codigoHash);
    }
}

package relaciones_1_a_1_ejercicio8;

public class Principal {

    public static void main(String[] args) {

        Documento doc1 = new Documento ("Contrato de locacion", "alquiler propiedad",
"#456hhy", "12/12/25");

        Usuario u1 = new Usuario("Pedro", "pedro@hotmail.com");

        FirmaDigital f1 =new FirmaDigital("#456hhy", "12/12/25");

        doc1.mostrarInformacion();

        u1.setFirmaDigital(f1);

        System.out.println(u1.acceder());

    }

}

```

Console:

run:

Titulo: Contrato de locacion

Mostrando firma digital: #456hhy

Hash: #456hhy, Fecha: 12/12/25

BUILD SUCCESSFUL (total time: 0 seconds)

9. CitaMédica - Paciente - Profesional

- a. Asociación unidireccional: [CitaMédica](#) → [Paciente](#),
- b. Asociación unidireccional: [CitaMédica](#) → [Profesional](#)

Clases y atributos:

i.

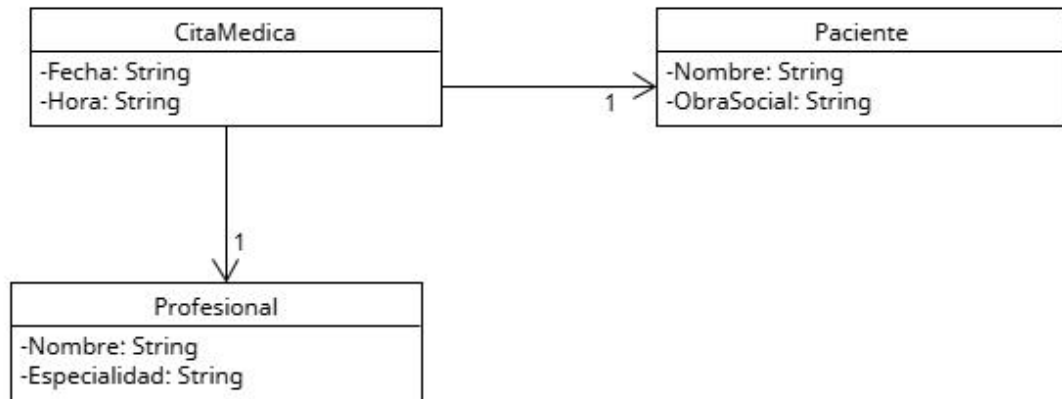
CitaMédica: fecha, hora

ii.

Paciente: nombre, obraSocial

iii.

Profesional: nombre, especialidad



a. Asociación unidireccional: **CitaMédica** → **Paciente**,

b. Asociación unidireccional: **CitaMédica** → **Profesional**

```
package relaciones_1_a_1_ejercicio9;
```

```
public class Principal {
```

```
    public static void main(String[] args) {
```

```
        Paciente p1 = new Paciente("Eugenia", "Omint");
```

```
        CitaMedica c1= new CitaMedica("22/11/25", "10.45");
```

```
        Profesional profesional1 = new Profesional ("Carla Dambrosio","Ginecologia");
```

```
        p1.setCitaMedica(c1);
```

```
        System.out.println(p1);
```

```
        System.out.println("La fecha de la cita del paciente " + p1.getNombre() + " es : " +  
p1.getFechaCitaMedica());
```

```
        profesional1.setCitaMedica(c1);
```

```
        System.out.println(profesional1);
```

```
    }
```

```
}
```

```

package relaciones_1_a_1_ejercicio9;

public class CitaMedica { //clase asociada por Paciente y por Profesional

    private String fecha;

    private String hora;


    public CitaMedica(String fecha, String hora){

        this.fecha = fecha;

        this.hora = hora;

    }


    public String getFecha() {

        return fecha;

    }


    public String getHora() {

        return hora;

    }

    @Override

    public String toString() {

        return "CitaMedica{" + "fecha=" + fecha + ", hora=" + hora + '}';

    }

}

```

```

package relaciones_1_a_1_ejercicio9;

public class Paciente {

    private String nombre;

    private String obraSocial;

    private CitaMedica citaMedica; //Atributo de la relacion


    public Paciente(String nombre, String obraSocial){

        this.nombre = nombre;

    }

```

```

        this.obraSocial = obraSocial;
    }
    public void setCitaMedica(CitaMedica citaMedica){
        this.citaMedica = citaMedica;
    }
    public String getFechaCitaMedica(){
        return citaMedica!= null ? citaMedica.getFecha() : "Sin fecha";
    }
    public String getNombre() {
        return nombre;
    }
    public String getObraSocial() {
        return obraSocial;
    }

    @Override
    public String toString() {
        return "Paciente{" + "nombre=" + nombre + ", obraSocial=" + obraSocial + ",
citaMedica=" + citaMedica + '}';
    }

```

```

package relaciones_1_a_1_ejercicio9;

```

```

public class Profesional {
    private String nombre;
    private String especialidad;
    private CitaMedica citaMedica;// atributo de la relacion

    public Profesional(String nombre, String especialidad){
        this.nombre = nombre;
        this.especialidad = especialidad;
    }

    public void setCitaMedica(CitaMedica citaMedica){
        this.citaMedica = citaMedica;
    }

    public String getCitaMedica(){
        return citaMedica != null ? citaMedica.getFecha() : "Sin cita medica";
    }

    @Override
    public String toString() {
        return "Profesional{" + "nombre=" + nombre + ", especialidad=" + especialidad +
", citaMedica=" + citaMedica + '}';
    }

```

```
}  
  
}
```

10. CuentaBancaria - ClaveSeguridad - Titular

a. Composición: **CuentaBancaria → ClaveSeguridad**

b. Asociación bidireccional: **CuentaBancaria ↔ Titular**

Clases y atributos:

i.

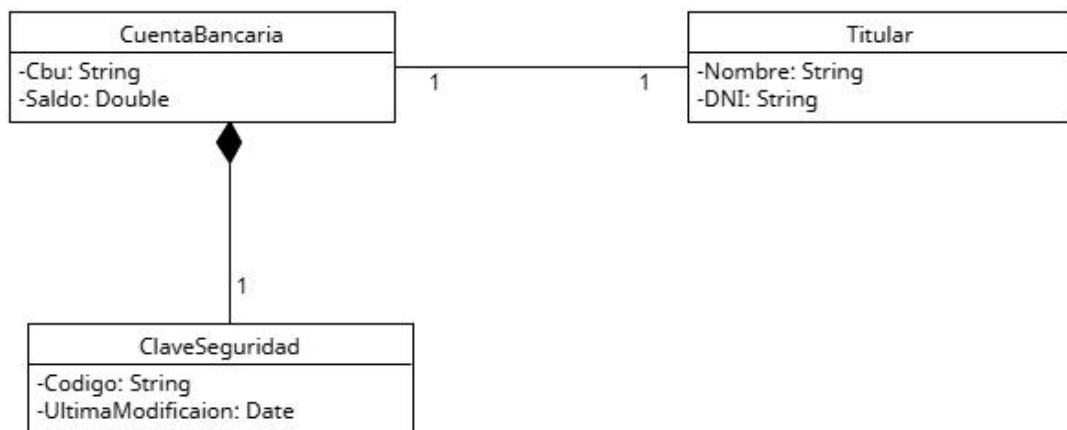
CuentaBancaria: cbu, saldo

ii.

ClaveSeguridad: codigo, ultimaModificacion

iii.

Titular: nombre, dni.



a. Composición: **CuentaBancaria → ClaveSeguridad**

```
package relaciones_1_a_1_ejercicio10;
```

```
public class CuentaBancaria {
```

```
    private String cbu;
```

```
    private String saldo;
```

```
    private ClaveSeguridad claveSeguridad;// atributo de composicion
```

```
    public CuentaBancaria(String cbu, String saldo, String codigo, String ultimaModificacion){
```

```
        this.cbu = cbu;
```

```

        this.saldo = saldo;

        //Creacion directa de la parte

        this.claveSeguridad = new ClaveSeguridad(codigo, ultimaModificacion);
    }


    //Getter para acceder a la parte
    public ClaveSeguridad getClaveSeguridad(){
        return claveSeguridad;
    }


    //FUNCION QUE UTILIZA LA PARTE:
    public void mostrarInformacion(){
        System.out.println("La cuenta con CBU: " + cbu);
        claveSeguridad.mostrar();
    }


    @Override
    public String toString() {
        return "CuentaBancaria{" + "cbu=" + cbu + ", saldo=" + saldo + ", claveSeguridad=" +
        claveSeguridad + '}';
    }
}

```

```

package relaciones_1_a_1_ejercicio10;

public class ClaveSeguridad {

    private String codigo;

    private String ultimaModificacion;

```

```
public ClaveSeguridad(String codigo, String ultimaModificacion){  
    this.codigo = codigo;  
    this.ultimaModificacion = ultimaModificacion;  
}
```

```
public String getCodigo() {  
    return codigo;  
}
```

```
public String getUltimaModificacion() {  
    return ultimaModificacion;  
}
```

```
public void mostrar(){  
    System.out.println("Mostrando clave: " + codigo );  
}
```

```
@Override
```

```
public String toString() {  
    return "ClaveSeguridad{" + "codigo=" + codigo + ", ultimaModificacion=" +  
ultimaModificacion + '}';  
}  
  
}
```

```
package relaciones_1_a_1_ejercicio10;
```

```
public class Principal {
```

```
public static void main(String[] args) {
```

```
    CuentaBancaria cuenta1 = new CuentaBancaria("82783823673672", "$9000000",  
"12345", "5/8/24");
```

```
cuenta1.mostrarInformacion();  
  
System.out.println(cuenta1);  
  
}  
}
```

Console:

La cuenta con CBU: 82783823673672

Mostrando clave: 12345

CuentaBancaria{cbu=82783823673672, saldo=\$90000000,
claveSeguridad=ClaveSeguridad{codigo=12345, ultimaModificacion=5/8/24}}

BUILD SUCCESSFUL (total time: 0 seconds)

b. Asociación bidireccional: CuentaBancaria ↔ Titular

```
package relaciones_1_a_1_ejercicio10;
```

```
public class Titular {  
  
    private String nombre;  
  
    private String DNI;  
  
    private CuentaBancaria cuentaBancaria;  
  
    public Titular(String nombre, String DNI) {  
        this.nombre = nombre;  
        this.DNI = DNI;  
    }  
}
```

```
//Funcion setter con validacion anti-bucle
```

```

public void setCuentaBancaria(CuentaBancaria cuentaBancaria){
    if (this.cuentaBancaria != cuentaBancaria){
        this.cuentaBancaria = cuentaBancaria;
        if (cuentaBancaria != null && cuentaBancaria.getTitular() != this){
            cuentaBancaria.setTitular(this);
        }
    }
}

```

//FUNCION GETTER PARA ACCEDER A DATOS RELACIONADOS:

```

public CuentaBancaria getCuentaBancaria(){
    return cuentaBancaria;
}

```

```

public String getCbuCuentaBancaria(){
    return cuentaBancaria != null ? cuentaBancaria.getCbu() : "Sin CBU";
}

```

```

public String getNombre() {
    return nombre;
}

```

@Override

```

public String toString() {
    return "Titular{" + "nombre=" + nombre + ", DNI=" + DNI + ", cuentaBancaria=" +
cuentaBancaria + '}';
}

```



```
}
```

```
package relaciones_1_a_1_ejercicio10;
```

```
public class CuentaBancaria {
```

```
    private String cbu;
```

```
    private String saldo;
```

```
    private ClaveSeguridad claveSeguridad;// atributo de composicion
```

```
    private Titular titular; // atributo de asociacion bidireccional
```

```
    public CuentaBancaria(String cbu, String saldo, String codigo, String  
ultimaModificacion){
```

```
        this.cbu = cbu;
```

```
        this.saldo = saldo;
```

```
        //Creacion directa de la parte
```

```
        this.claveSeguridad = new ClaveSeguridad(codigo, ultimaModificacion);
```

```
    }
```

```
    //Getter para acceder a la parte
```

```
    public ClaveSeguridad getClaveSeguridad(){
```

```
        return claveSeguridad;
```

```
    }
```

```
    //funcion QUE UTILIZA LA PARTE:
```

```
    public void mostrarInformacion(){
```

```
        System.out.println("La cuenta con CBU: " + cbu);
```

```
        claveSeguridad.mostrar();
```

```
    }
```

```

public String getCbu() {
    return cbu;
}

```

//Para la asociacion bidireccional: funcion setter con validacion antibucle:

```

public void setTitular(Titular titular){
    if (this.titular != titular){
        this.titular = titular;
        if (titular != null && titular.getCuentaBancaria() != this){
            titular.setCuentaBancaria(this);
        }
    }
}

```

//Funciopn getter para acceder a datos relacionados

```

public Titular getTitular(){
    return titular;
}

```

```

public String getNombreTitular(){
    return titular !=null ? titular.getNombre() : "Sin nombre";
}

```

@Override

```

public String toString() {
    return "CuentaBancaria{" + "cbu=" + cbu + ", saldo=" + saldo + ", claveSeguridad="
+ claveSeguridad + '}';
}
}

```

```
package relaciones_1_a_1_ejercicio10;
```

```
public class Principal {  
    public static void main(String[] args) {  
        CuentaBancaria cuenta1 = new CuentaBancaria("82783823673672", "$9000000",  
"12345", "5/8/24");  
  
        Titular t1 = new Titular("Pepe Luis", "23777888");  
  
        cuenta1.mostrarInformacion();  
  
        System.out.println(cuenta1);  
  
        //para ver la bidireccionalidad uso una de las funciones setter  
  
        t1.setCuentaBancaria(cuenta1);  
  
        System.out.println(t1);  
    }  
}
```

DEPENDENCIA DE USO

La clase usa otra como **parámetro de un método**, pero **no la guarda como atributo**.

Ejercicios de Dependencia de Uso

11. Reproductor - Canción - Artista

- a. Asociación unidireccional: **Canción → Artista**
- b. Dependencia de uso: **Reproductor.reproducir(Cancion)**

Clases y atributos:

i.

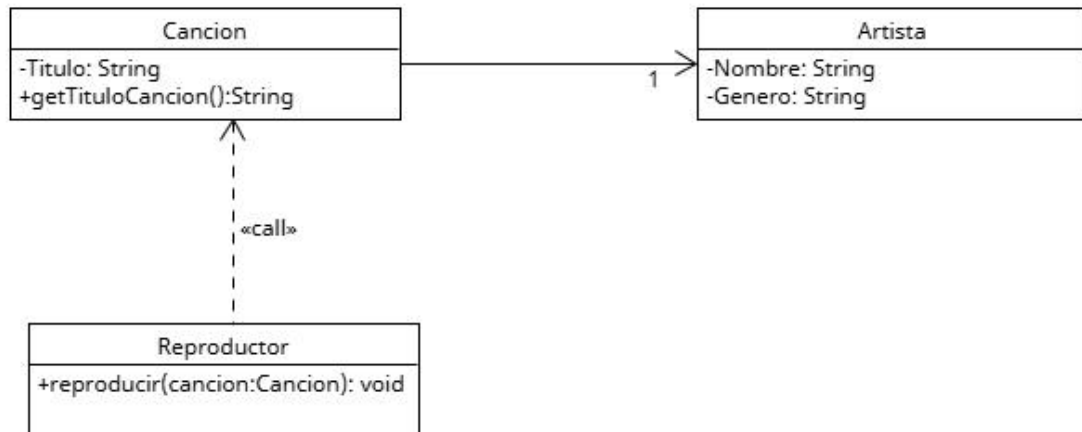
Canción: titulo.

ii.

Artista: nombre, genero.

iii.

Reproductor->método: void reproducir(Cancion cancion)



a. Asociación unidireccional: Canción → Artista

```
package Dependencia_Uso.ejercicio11;
```

```
public class Cancion { //en asociacion 1 a 1 es la clase que mantiene la referencia
```

```
    private String titulo;
```

```
    private Artista artista;
```

```
    public Cancion(String titulo) {
```

```
        this.titulo = titulo;
```

```
    }
```

```
    public void setArtista(Artista artista){
```

```
        this.artista = artista;
```

```
    }
```

```
//Funcion getter para acceder a datos relacionados
```

```
public String getNombreArtista(){
```

```
    return artista != null ? artista.getNombre() : "Sin nombre";
```

```
}
```

```
public String getGeneroArtista(){
```

```
    return artista != null ? artista.getGenero(): "Sin genero";
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    return "Cancion{" + "titulo=" + titulo + ", artista=" + artista + '}';
```

```
}
```

```
}
```

```
package Dependencia_Uso.ejercicio11;
```

```
public class Artista {
```

```
    private String nombre;
```

```
    private String genero;
```

```
    public Artista(String nombre, String genero) {
```

```
        this.nombre = nombre;
```

```
        this.genero = genero;
```

```
}
```

```
public String getNombre() {  
    return nombre;  
}
```

```
public String getGenero() {  
    return genero;  
}
```

```
@Override  
public String toString() {  
    return "Artista{" + "nombre=" + nombre + ", genero=" + genero + '}';  
}
```

```
}
```

```
package Dependencia_Uso.ejercicio11;  
  
public class Principal {  
    public static void main(String[] args) {  
        Cancion cancion = new Cancion ("Memory Gospel");  
        Artista artista = new Artista("Moby", "electronica");  
  
        cancion.setArtista(artista);  
        System.out.println(cancion);  
    }  
}
```

```
}
```

```
}
```

Consola:

run:

Cancion{titulo=Memory Gospel, artista=Artista{nombre=Moby, genero=electronica}}

BUILD SUCCESSFUL (total time: 0 seconds)

b. Dependencia de uso: `Reproductor.reproducir(Cancion)`

```
package Dependencia_Uso.ejercicio11;
```

```
public class Reproductor {
```

```
//No hay atributo de reproductor
```

```
//Metodo que usa temporalmente una Cancion (dependencia de uso)
```

```
public void reproducir(Cancion cancion){
```

```
    System.out.println("Titulo de la cancion: " + cancion.getTituloCancion());
```

```
    System.out.println("Artista: " + cancion.getNombreArtista());
```

```
    //La cancion se usa solo durante la ejecucion del metodo
```

```
}
```

```
}
```

```
package Dependencia_Uso.ejercicio11;

public class Cancion { //en asociacion 1 a 1 es la clase que mantiene la referencia

    private String titulo;

    private Artista artista;


    public Cancion(String titulo) {

        this.titulo = titulo;

    }


    public void setArtista(Artista artista){

        this.artista = artista;

    }


    //Funcion getter para acceder a datos relacionados

    public String getNombreArtista(){

        return artista != null ? artista.getNombre() : "Sin nombre";

    }

    public String getGeneroArtista(){

        return artista != null ? artista.getGenero(): "Sin genero";

    }

    //DEPENDENCIA DE USO


    public String getTituloCancion(){

        return titulo;

    }

}
```



```
public Artista getArtista() {  
    return artista;  
}
```

```
@Override  
public String toString() {  
    return "Cancion{" + "titulo=" + titulo + ", artista=" + artista + '}';  
}  
}
```

```
package Dependencia_Uso.ejercicio11;
```

```
public class Principal {  
    public static void main(String[] args) {  
        Cancion cancion = new Cancion ("Memory Gospel");  
        Artista artista = new Artista("Moby", "electronica");  
  
        cancion.setArtista(artista);  
  
        System.out.println(cancion);  
        Reproductor reproductor = new Reproductor ();  
  
        //Usar la dependencia temporalmente
```

```
reproductor.reproducir(cancion);
```

```
}
```

```
}
```

Console:

run:

Cancion{titulo=Memory Gospel, artista=Artista{nombre=Moby, genero=electronica}}

Titulo de la cancion: Memory Gospel

Artista: Moby

BUILD SUCCESSFUL (total time: 0 seconds)

12. Impuesto - Contribuyente - Calculadora

a. Asociación unidireccional: **Impuesto → Contribuyente**

b. Dependencia de uso: **Calculadora.calcular(Impuesto)**

Clases y atributos:

i.

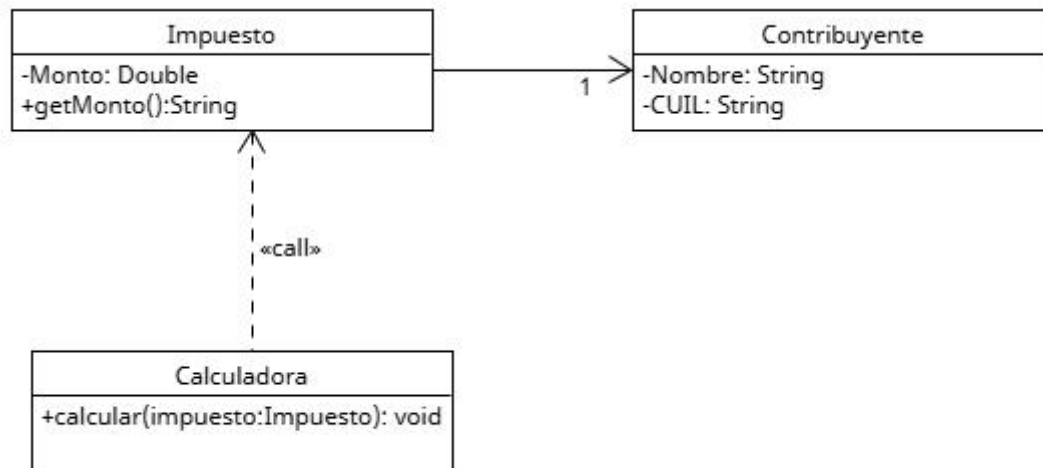
Impuesto: monto.

ii.

Contribuyente: nombre, cuil.

iii.

Calculadora->método: void calcular(Impuesto impuesto)



a. Asociación unidireccional: **Impuesto** → **Contribuyente**

```
package Dependencia_Uso.ejercicio12;
```

```
public class Impuesto {//asociacion 1 a 1 con Contribuyente
```

```
private double monto;
```

```
private Contribuyente contribuyente;
```

```
public Impuesto(double monto) {
    this.monto = monto;
}
```

```
public void setContribuyente(Contribuyente contribuyente){
    this.contribuyente =contribuyente;
}
```

```
//Funcion getter para acceder a datos relacionados
```

```
public String getNombreContribuyente(){
    return contribuyente != null ? contribuyente.getNombre() : "Sin nombre";
}
```

```
@Override
```

```
public String toString() {  
    return "Impuesto{" + "monto=" + monto + ", contribuyente=" + contribuyente + '}';  
}  
}
```

```
package Dependencia_Uso.ejercicio12;
```

```
public class Contribuyente {
```

```
    private String nombre;
```

```
    private String cuil;
```

```
    public Contribuyente(String nombre, String cuil) {
```

```
        this.nombre = nombre;
```

```
        this.cuil = cuil;
```

```
    }
```

```
    public String getNombre() {
```

```
        return nombre;
```

```
    }
```

```
    public String getCuil() {
```

```
        return cuil;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Contribuyente{" + "nombre=" + nombre + ", cuil=" + cuil + '}';
```

```
    }
```

```
}
```

```
package Dependencia_Uso.ejercicio12;

public class Principal {

    public static void main(String[] args) {

        Impuesto impuesto = new Impuesto(9000);

        Contribuyente c1 = new Contribuyente("Fernando Luis", "28372389738-9");

        impuesto.setContribuyente(c1);

        System.out.println(impuesto);

    }

}
```

Console:

run:

```
Impuesto{monto=90.0, contribuyente=Contribuyente{nombre=Fernando Luis,
cuil=28372389738-9}}
```

BUILD SUCCESSFUL (total time: 0 seconds)

b. Dependencia de uso: [Calculadora.calcular\(Impuesto\)](#)

```
package Dependencia_Uso.ejercicio12;

public class Calculadora {

    //metodo que usa temporalmente el impuesto

    public void calcular(Impuesto impuesto){

        System.out.println("El monto del impuesto es de: " + impuesto.getMonto());

        System.out.println("Sera deducido del contribuyente: " +
impuesto.getNombreContribuyente());

    }

}
```

```
package Dependencia_Uso.ejercicio12;

public class Impuesto { //asociacion 1 a 1 con Contribuyente

    private double monto;

    private Contribuyente contribuyente;


    public Impuesto(double monto) {

        this.monto = monto;

    }


    public void setContribuyente(Contribuyente contribuyente){

        this.contribuyente =contribuyente;

    }

    //Funcion getter para acceder a datos relacionados


    public String getNombreContribuyente(){

        return contribuyente != null ? contribuyente.getNombre() : "Sin nombre";

    }


    //getters para que el metodo calcular los llame


    public double getMonto() {

        return monto;

    }


    public Contribuyente getContribuyente() {

        return contribuyente;

    }

    @Override
```

```
public String toString() {  
    return "Impuesto{" + "monto=" + monto + ", contribuyente=" + contribuyente + '}';  
}  
}
```

```
package Dependencia_Uso.ejercicio12;  
  
public class Principal {  
    public static void main(String[] args) {  
        Impuesto impuesto = new Impuesto(9000);  
        Contribuyente c1 = new Contribuyente("Fernando Luis", "28372389738-9");  
        Calculadora calcu = new Calculadora();  
  
        impuesto.setContribuyente(c1);  
        System.out.println(impuesto);  
  
        calcu.calcular(impuesto);  
    }  
}
```

Console:

```
Impuesto{monto=9000.0, contribuyente=Contribuyente{nombre=Fernando Luis,  
cuil=28372389738-9}}
```

El monto del impuesto es de: 9000.0

Sera deducido del contribuyente: Fernando Luis

BUILD SUCCESSFUL (total time: 0 seconds)

DEPENDENCIA DE CREACIÓN

La clase crea otra dentro de un método, pero no la conserva como atributo..

Ejercicios de Dependencia de Creación

13. GeneradorQR - Usuario - CódigoQR

a. Asociación unidireccional: **CódigoQR → Usuario**

b. Dependencia de creación: **GeneradorQR.generar(String, Usuario)**

Clases y atributos:

i.

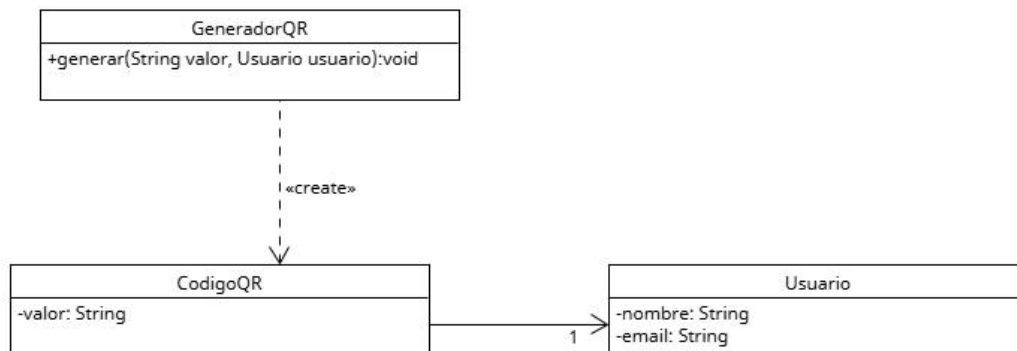
CodigoQR: valor.

ii.

Usuario: nombre, email.

iii.

GeneradorQR->método: void generar(String valor, Usuario usuario)



a. Asociación unidireccional: **CódigoQR → Usuario**

```
package Dependencia_Creacion.ejercicio_13;
```



```
public class CodigoQR {  
    private String valor;  
    private Usuario usuario;  
  
    public CodigoQR(String valor, Usuario usuario){  
        this.valor = valor;  
        this.usuario = usuario;  
    }  
  
    public String getValor(){  
        return valor;  
    }  
  
    public Usuario getUsuario() {  
        return usuario;  
    }  
  
    //metodo para la validacion de la creacion:  
    public void guardarEnBD(){  
        System.out.println("codigo guardado: " + valor);  
    }  
  
    //Funcion setter para establecer la relacion:  
    public void setUsuario(Usuario usuario){  
        this.usuario = usuario;  
    }  
}
```

```
//getter para acceder a datos relacionados
public String getNombre(){
    return usuario != null ? usuario.getNombre() : "Sin pasaporte";
}

@Override
public String toString() {
    return "CodigoQR{" + "valor=" + valor + ", usuario=" + usuario + '}';
}
}
```

```
package Dependencia_Creacion.ejercicio_13;

public class Usuario {
    private String nombre;
    private String email;
```

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

```
    public String getNombre() {
        return nombre;
    }
```

```
public String getEmail() {  
    return email;  
}
```

@Override

```
public String toString() {  
    return "Usuario{" + "nombre=" + nombre + ", email=" + email + '}';  
}  
}
```

```
package Dependencia_Creacion.ejercicio_13;
```

```
public class Principal {
```

```
    public static void main(String[] args) {
```

```
        Usuario usuario = new Usuario("Luis Kike", "kike@gmail.com");
```

```
        Usuario usuarioParaCreacion = new Usuario("Luisa Li", "luli@gmail.com");
```

```
        CodigoQR codigo = new CodigoQR("888", usuario);
```

```
        //establezco la relacion unidireccional:
```

```
        codigo.setUsuario(usuario);
```

```
        System.out.println(codigo);
```

```
        //implementacion de dependencia de Creacion:
```

```
        //Crear generador
```

```
        GeneradorQR generador = new GeneradorQR();
```

```

        //generar codigo

        CodigoQR codigo2 = generador.generar("111", usuarioParaCreacion);


        System.out.println(codigo2);

    }
}

```

b. Dependencia de creación: **GeneradorQR.generar(String, Usuario)**

```

package Dependencia_Creacion.ejercicio_13;


public class GeneradorQR {

    //no hay atributo codigoQR


    // Dependencia de creación: Genera un CodigoQR TEMPORAL


        public CodigoQR generar(String valor, Usuario usuario) {

            CodigoQR nuevoCodigo = new CodigoQR(valor, usuario);


            //Validacion
            if (validarCodigo(nuevoCodigo)){

                nuevoCodigo.guardarEnBD();

                System.out.println("Codigo creado exitosamente");

                return nuevoCodigo;

            }

            //el objeto nuevoCodigo SE DESTRUYE al finalizar el metodo

            System.out.println("Error: no se puede crear el codigo");

```

```
return null;
```

```
}
```

```
private boolean validarCodigo(CodigoQR codigo){
```

```
    return codigo.getValor() != null && !codigo.getValor().isEmpty();
```

```
}
```

```
}
```

```
package Dependencia_Creacion.ejercicio_13;
```

```
public class CodigoQR {
```

```
    private String valor;
```

```
    private Usuario usuario;
```

```
    public CodigoQR(String valor, Usuario usuario){
```

```
        this.valor = valor;
```

```
        this.usuario = usuario;
```

```
}
```

```
    public String getValor(){
```

```
        return valor;
```

```
}
```

```
    public Usuario getUsuario() {
```

```
        return usuario;
```

```
}
```

```
//metodo para la validacion de la creacion:
```

```
public void guardarEnBD(){  
    System.out.println("codigo guardado: " + valor);  
}
```

//Funcion setter para establecer la relacion:

```
public void setUsuario(Usuario usuario){  
    this.usuario = usuario;  
}
```

//getter para acceder a datos relacionados

```
public String getNombre(){  
    return usuario != null ? usuario.getNombre() : "Sin pasaporte";  
}
```

@Override

```
public String toString() {  
    return "CodigoQR{" + "valor=" + valor + ", usuario=" + usuario + "}";  
}  
}
```

```
package Dependencia_Creacion.ejercicio_13;
```

```
public class Principal {
```

```
    public static void main(String[] args) {
```

```
        Usuario usuario = new Usuario("Luis Kike", "kike@gmail.com");
```

```
        Usuario usuarioParaCreacion = new Usuario("Luisa Li", "luli@gmail.com");
```

```
        CodigoQR codigo = new CodigoQR("888", usuario);
```

```

//establezco la relacion unidireccional:

codigo.setUsuario(usuario);

System.out.println(codigo);


//implementacion de dependencia de Creacion:

//Crear generador

GeneradorQR generador = new GeneradorQR();


//generar codigo

CodigoQR codigo2 = generador.generar("111", usuarioParaCreacion);


System.out.println(codigo2);


}
}

```

14. EditorVideo - Proyecto - Render

a. Asociación unidireccional: **Render → Proyecto**

b. Dependencia de creación: **EditorVideo.exportar(String, Proyecto)**

c. Clases y atributos:

i.

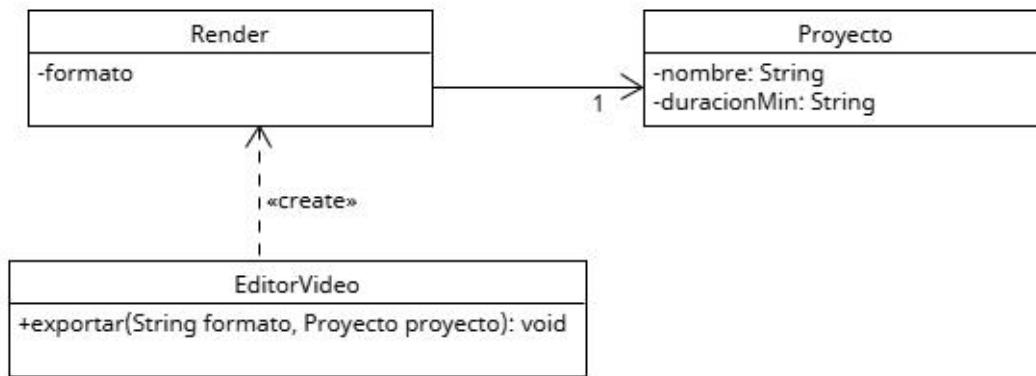
Render: formato.

ii.

Proyecto: nombre, duracionMin.

iii.

EditorVideo->método: void exportar(String formato, Proyecto proyecto)



a. Asociación unidireccional: **Render** → **Proyecto**

```
package Dependencia_Creacion.ejercicio_14;
```

```
public class Render {
```

```
    private Proyecto proyecto;
```

```
    public void setProyecto(Proyecto proyecto){
```

```
        this.proyecto = proyecto;
```

```
    }
```

```
    public String getNombreProyecto(){
```

```
        return proyecto != null ? proyecto.getNombre() : "Sin nombre";
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Render{" + "proyecto=" + proyecto + '}';
```

```
    }
```

```
}
```



```
package Dependencia_Creacion.ejercicio_14;
```

```
public class Proyecto {
```

```
    private String nombre;
```

```
    private String duracionMin;
```

```
    public Proyecto(String nombre, String duracionMin){
```

```
        this.nombre= nombre;
```

```
        this.duracionMin = duracionMin;
```

```
    }
```

```
    public String getNombre() {
```

```
        return nombre;
```

```
    }
```

```
    public String getDuracionMin() {
```

```
        return duracionMin;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Proyecto{" + "nombre=" + nombre + ", duracionMin=" + duracionMin + '}';
```

```
    }
```

```
}
```

```
}
```

```
package Dependencia_Creacion.ejercicio_14;
```

```
public class Principal {  
    public static void main(String[] args) {  
        Proyecto p1 = new Proyecto("cumpleaños", "15");  
        Render r1 = new Render();  
        r1.setProyecto(p1);  
        System.out.println(r1);  
    }  
}
```

Consola:

run:

Render{proyecto=Proyecto{nombre=cumpleaños, duracionMin=15}}

BUILD SUCCESSFUL (total time: 0 seconds)

b. Dependencia de creación: `EditorVideo.exportar(String, Proyecto)`

```
package Dependencia_Creacion.ejercicio_14;
```

```
public class EditorVideo {  
  
    public void exportar(String formato, Proyecto proyecto){  
        Render render = new Render();  
        render.setProyecto(proyecto, formato);  
        System.out.println("Exportando " + render);  
    }  
}
```

```
}
```

```
}
```

```
package Dependencia_Creacion.ejercicio_14;
```

```
public class Principal {
```

```
public static void main(String[] args) {
```

```
    Proyecto p1 = new Proyecto("cumpleaños", "15");
```

```
    Render r1 = new Render();
```

```
    EditorVideo editor = new EditorVideo();
```

```
    r1.setProyecto(p1, "MP4");
```

```
    System.out.println(r1);
```

```
    editor.exportar("MP4", p1);
```

```
}
```

```
}
```

Consola:

run:

Render{proyecto=cumplea💎os, duracion=15', formato=MP4}

Exportando Render{proyecto=cumplea💎os, duracion=15', formato=MP4}

BUILD SUCCESSFUL (total time: 0 seconds)