# Trabajo Practico N°5

# de

# Programación II

Estudiante: Tobias Leiva

Unidad: UML Básico

Universidad Tecnológica Nacional

**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

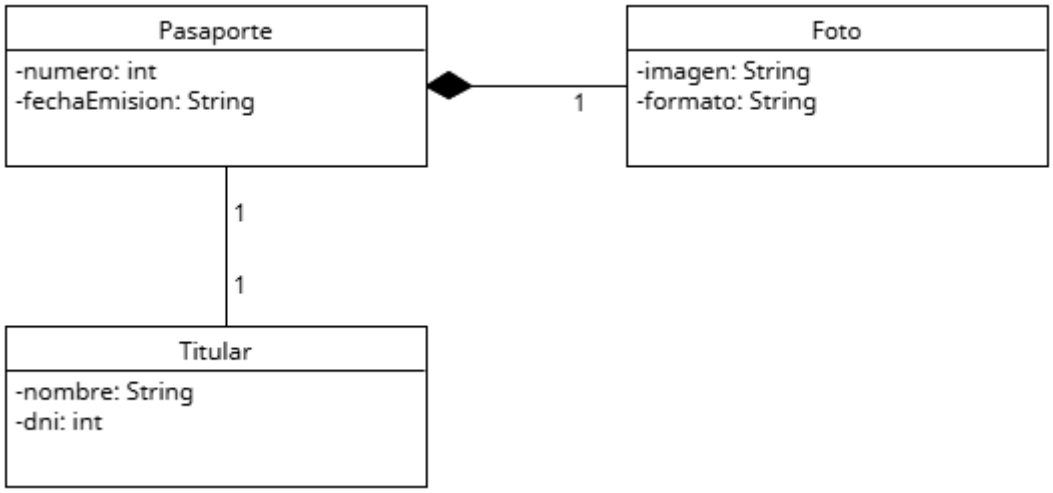
    ii. Foto: imagen, formato

    iii. Titular: nombre, dni

Las Clases en Java:

```
public class Pasaporte {
    private int numero;
    private String fechaEmision;
    private Titular titular; //Asociacion bidireccional

    public Pasaporte(int numero, String fechaEmision) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        Foto foto = new Foto(); //Composicion
    }
}
public class Foto {
    private String imagen;
    private String formato;
}
```

```
public class Titular {
    private String nombre;
    private int dni;
    private Pasaporte pasaporte; //Asociacion bidireccional
}
```




2. Celular - Batería - Usuario

a. Agregación: Celular → Batería

b. Asociación bidireccional: Celular ↔ Usuario

Clases y atributos:

i. Celular: imei, marca, modelo

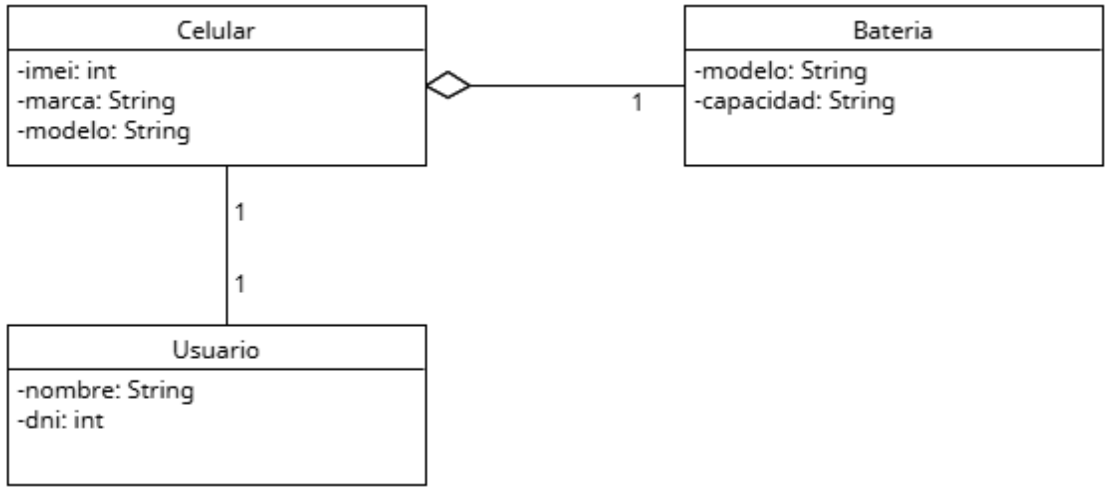
ii. Batería: modelo, capacidad

iii. Usuario: nombre, dni

Las Clases en Java:

```
public class Celular {
    private int imei;
    private String marca;
    private String modelo;
    private Bateria bateria; //Agregacion
    private Usuario usuario; //Asociacion bidireccional
}
public class Bateria {
    private String modelo;
    private double capacidad;
```

```
}
public class Usuario {
    private String nombre;
    private int dni;
    private Celular celular;
}
```




3. Libro - Autor - Editorial

a. Asociación unidireccional: Libro → Autor

b. Agregación: Libro → Editorial

Clases y atributos:

i. Libro: titulo, isbn

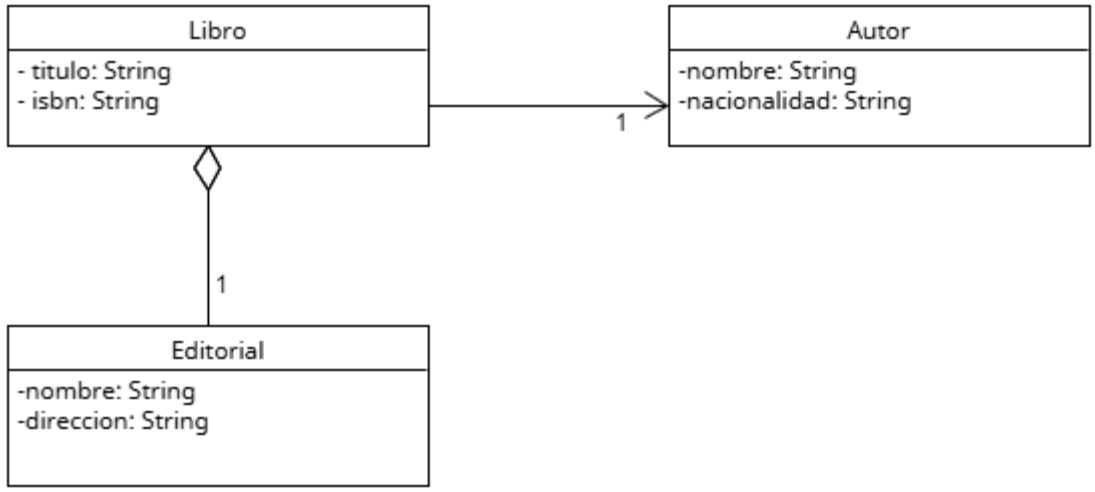
ii. Autor: nombre, nacionalidad

iii. Editorial: nombre, dirección

Las Clases en Java:

```
public class Libro {
    private String titulo;
    private String isbn;
    private Autor autor; //Asociacion unidireccional
    private Editorial editorial; //Agregacion 1:1
}
public class Editorial {
    private String nombre;
    private String direccion;
```

```
}
public class Autor {
    private String nombre;
    private String nacionalidad;
}
```




4. TarjetaDeCrédito - Cliente - Banco

a. Asociación bidireccional: TarjetaDeCrédito ↔ Cliente

b. Agregación: TarjetaDeCrédito → Banco

Clases y atributos:

i. TarjetaDeCrédito: numero, fechaVencimiento

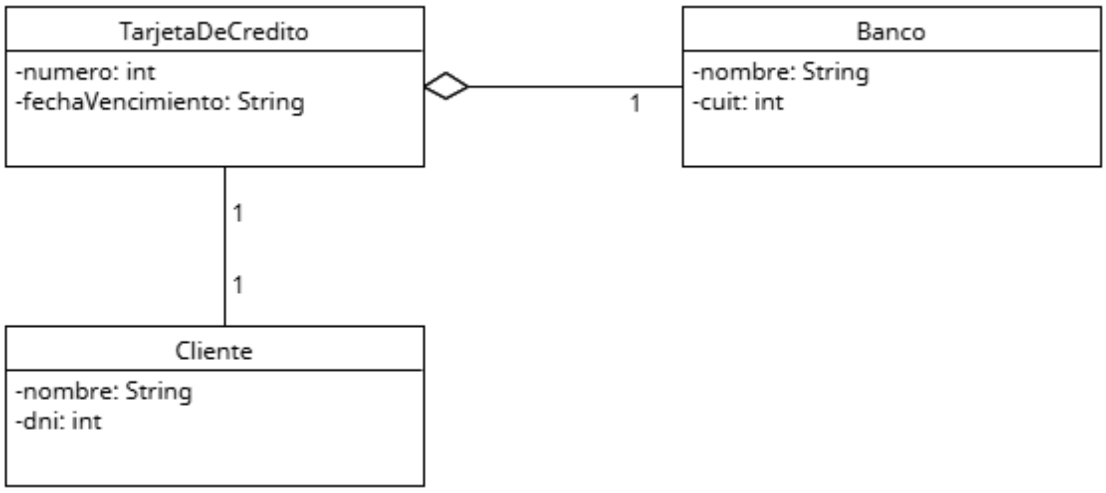
ii. Cliente: nombre, dni

iii. Banco: nombre, cuit

Las clases en Java:

```
public class TarjetaDeCredito {
    private int numero;
    private String fechaVencimiento;
    private Cliente cliente; //Asociacion bidireccional
    private Banco banco; //Agregacion 1:1
}
public class Cliente {
    private String nombre;
    private int dni;
```

```
    private TarjetaDeCredito tarjetaCredito; //Asociccion
bidireccional

}

public class Banco {

    private String nombre;

    private int cuit;

}
```




5. Computadora - PlacaMadre - Propietario

a. Composición: Computadora → PlacaMadre

b. Asociación bidireccional: Computadora ↔ Propietario

Clases y atributos:

i. Computadora: marca, numeroSerie

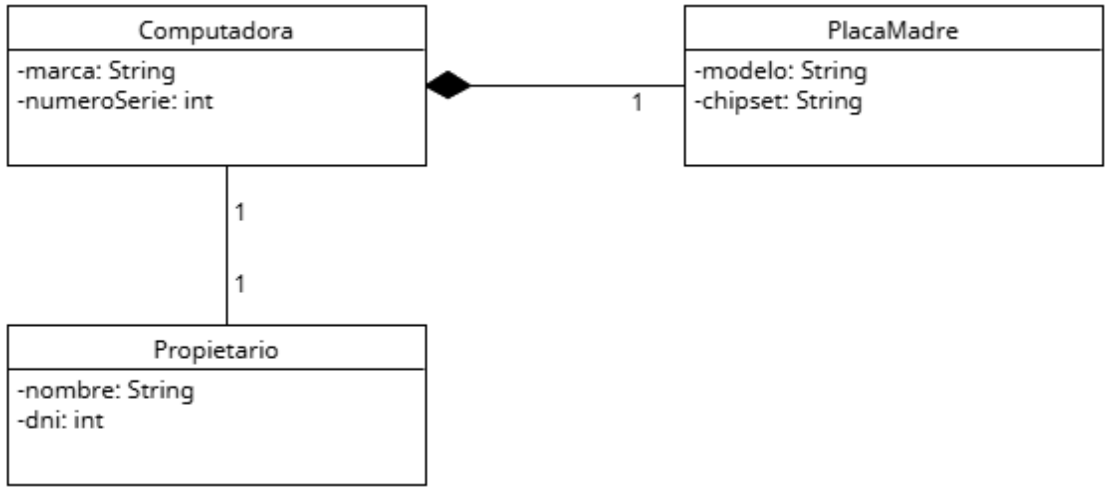
ii. PlacaMadre: modelo, chipset

iii. Propietario: nombre, dni

Las clases en Java:

```
public class Computadora {

    private String marca;

    private int numeroSerie;

    private Propietario propietario; //Asociacion bidireccional


    public Computadora(String marca, int numeroSerie, Propietario
propietario) {

        this.marca = marca;

        this.numeroSerie = numeroSerie;
```

```
        this.propietario = propietario;
        PlacaMadre placaMadre = new PlacaMadre(); //Composicion
    }

public class PlacaMadre {
    private String modelo;
    private String chipset;
}
public class Propietario {
    private String nombre;
    private int dni;
    private Computadora computadora; //Asociacion bidireccional
}
```




6. Reserva - Cliente - Mesa

    a. Asociación unidireccional: Reserva → Cliente

    b. Agregación: Reserva → Mesa

Clases y atributos:

    i. Reserva: fecha, hora

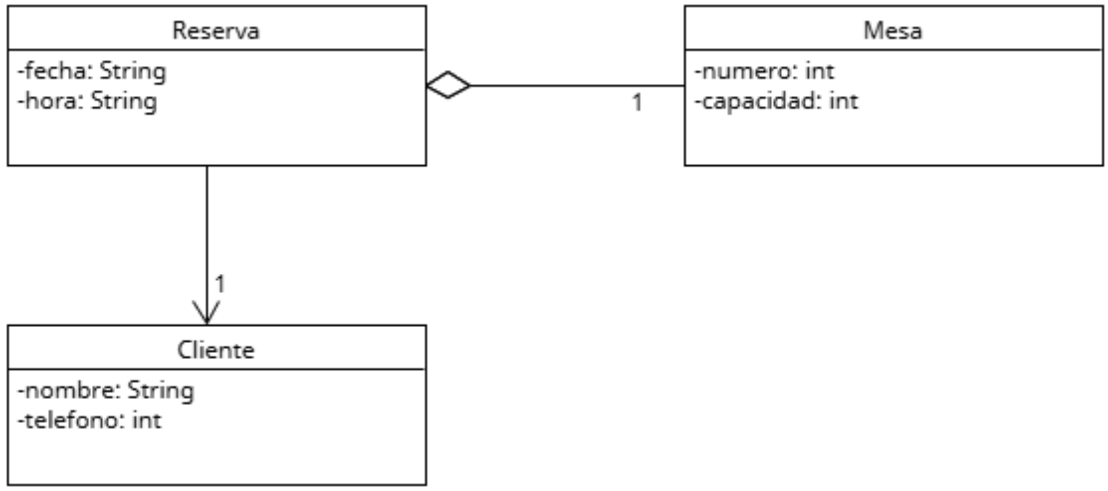
    ii. Cliente: nombre, telefono

    iii. Mesa: numero, capacidad

Las clases en Java:

```
public class Reserva {
    private String fecha;
```

```
    private String hora;
    private Cliente cliente; //Asociacion unidireccional
    private Mesa mesa; //Agregacion
}
public class Mesa {
    private int numero;
    private int capacidad;
}
public class Cliente {
    private String nombre;
    private int telefono;
}
```




7. Vehículo - Motor - Conductor

a. Agregación: Vehículo → Motor

b. Asociación bidireccional: Vehículo ↔ Conductor

Clases y atributos:

i. Vehículo: patente, modelo

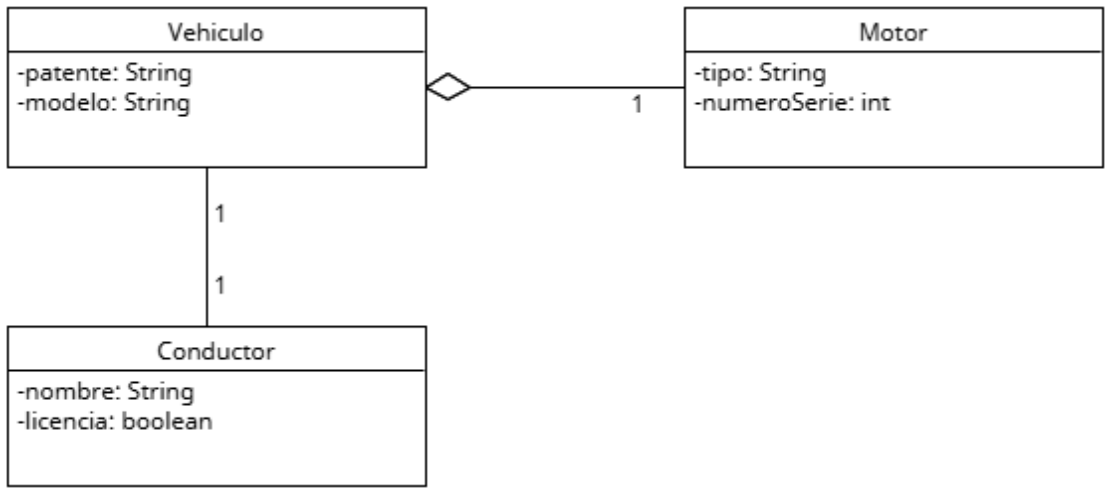
ii. Motor: tipo, numeroSerie

iii. Conductor: nombre, licencia

Las clases en Java:

```
public class Vehiculo {
    private String patente;
    private String modelo;
```

```
    private Motor motor; //Agregacion
    private Conductor conductor; //Asociacion bidireccional
}
public class Conductor {
    private String nombre;
    private boolean licencia;
    private Vehiculo vehiculo; //Asociacion bidireccional
}
public class Motor {
    private String tipo;
    private int numeroSerie;
}
```




8. Documento - FirmaDigital - Usuario

    a. Composición: Documento → FirmaDigital

    b. Agregación: FirmaDigital → Usuario

Clases y atributos:

    i. Documento: titulo, contenido

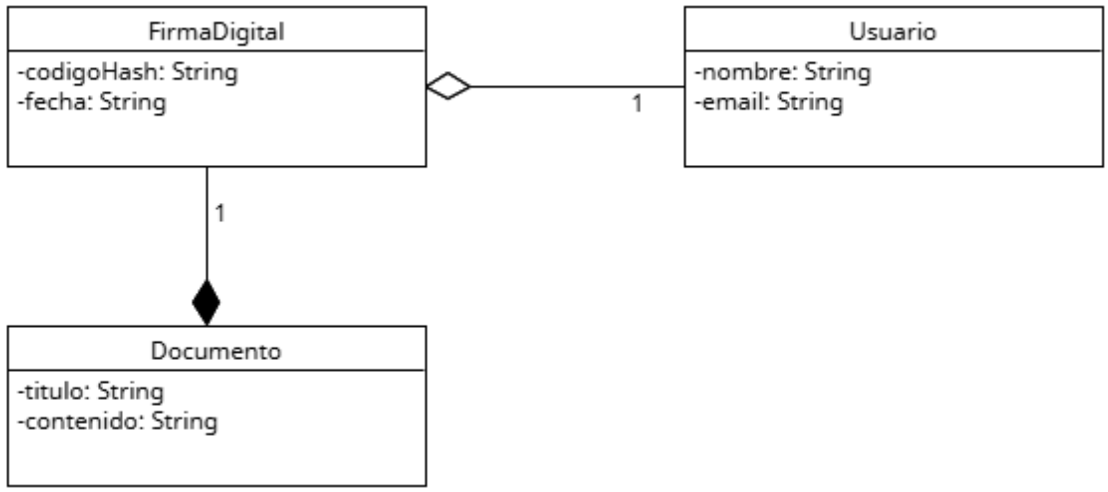
    ii. FirmaDigital: codigoHash, fecha

    iii. Usuario: nombre, email

Las clases en Java:

```
public class Documento {
    private String nombre;
    private String contenido;
```

```
    public Documento(String nombre, String contenido) {
        this.nombre = nombre;
        this.contenido = contenido;
        FirmaDigital firma = new FirmaDigital(); //Composicion
    }
}
public class FirmaDigital {
    private String codigoHash;
    private String fecha;
    private Usuario usuario; //Agregacion
}
public class Usuario {
    private String nombre;
    private String email;
}
```




9. CitaMédica - Paciente - Profesional

   a. Asociación unidireccional: CitaMédica → Paciente,

   b. Asociación unidirecciona: CitaMédica → Profesional

Clases y atributos:

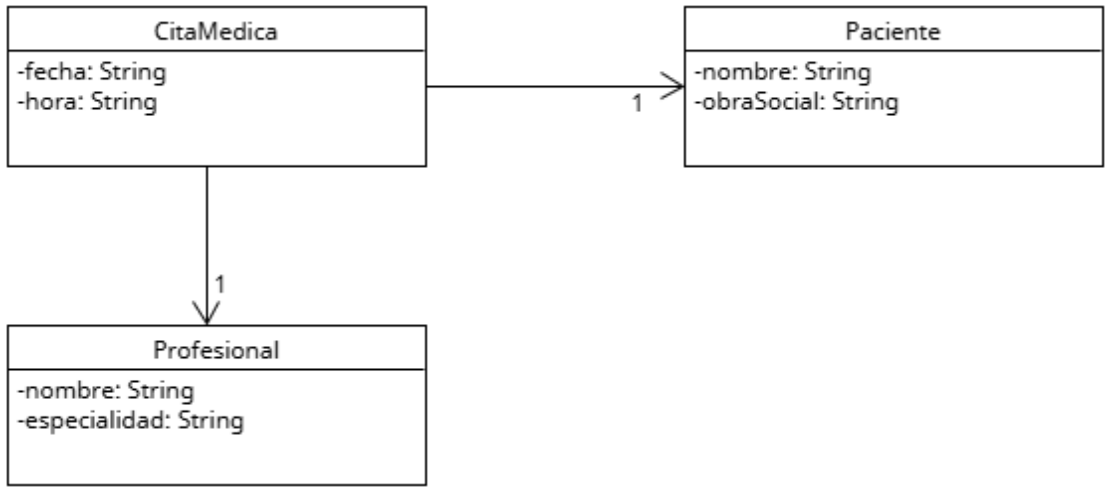
   i. CitaMédica: fecha, hora

   ii. Paciente: nombre, obraSocial

   iii. Profesional: nombre, especialidad

Las clases en Java:

```
public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente; //Asociacion unidireccional
    private Profesional profecional; //Asociacion unidireccional
}
public class Paciente {
    private String nombre;
    private String obraSocial;
}
public class Profesional {
    private String nombre;
    private String especialidad;
}
```




10. CuentaBancaria - ClaveSeguridad - Titular

    a. Composición: CuentaBancaria → ClaveSeguridad

    b. Asociación bidireccional: CuentaBancaria ↔ Titular

Clases y atributos:

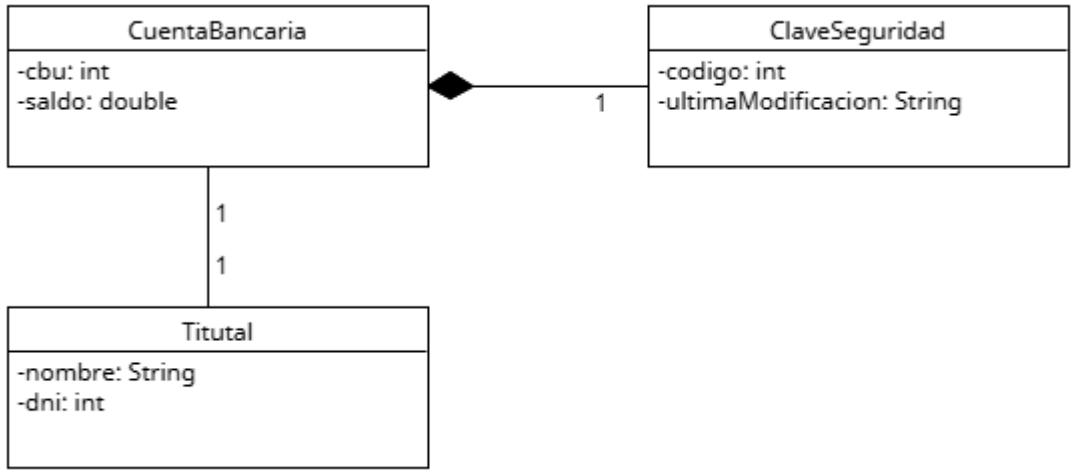
    i. CuentaBancaria: cbu, saldo

    ii. ClaveSeguridad: codigo, ultimaModificacion

    iii. Titular: nombre, dni.

Las clases en Java:

```
public class CuentaBancaria {
    private int cbu;
    private double saldo;
    private Titular titular; //Asociacion bidireccional

    public CuentaBancaria(int cbu, double saldo, Titular titular) {
        this.cbu = cbu;
        this.saldo = saldo;
        this.titular = titular;
        ClaveSeguridad clave = new ClaveSeguridad(); //Composicion
    }
}
public class ClaveSeguridad {
    private int codigo;
    private String ultimaModificacion;
}
public class Titular {
    private String nombre;
    private int dni;
    private CuentaBancaria cuentaBancaria; //Asociacion Bidireccional
}
```



**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.

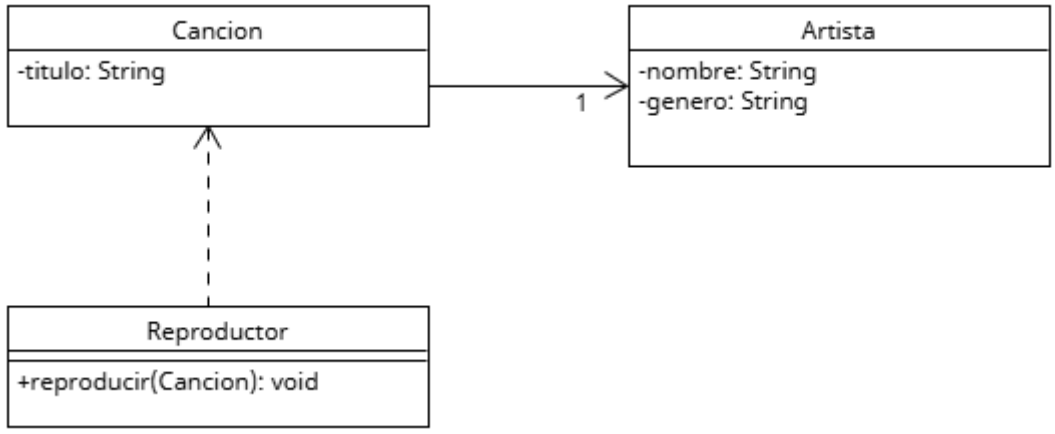
ii. Artista: nombre, genero.

iii. Reproductor->método: void reproducir(Cancion cancion)

Las clases en Java:

```
public class Cancion {
    private String titulo;
    private Artista artista; //Asociacion unidireccional

    public String getTitulo() {
        return titulo;
    }
    public Artista getArtista() {
        return artista;
    }
}
public class Reproductor {

    // Dependencia de uso: usa Cancion en un método
    public void reproducir(Cancion cancion){
        System.out.println("Reproduciendo: "
            + cancion.getTitulo()
            + " - " + cancion.getArtista().getNombre());
    }
}
```

```
public class Artista {
    private String nombre;
    private String genero;

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




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.

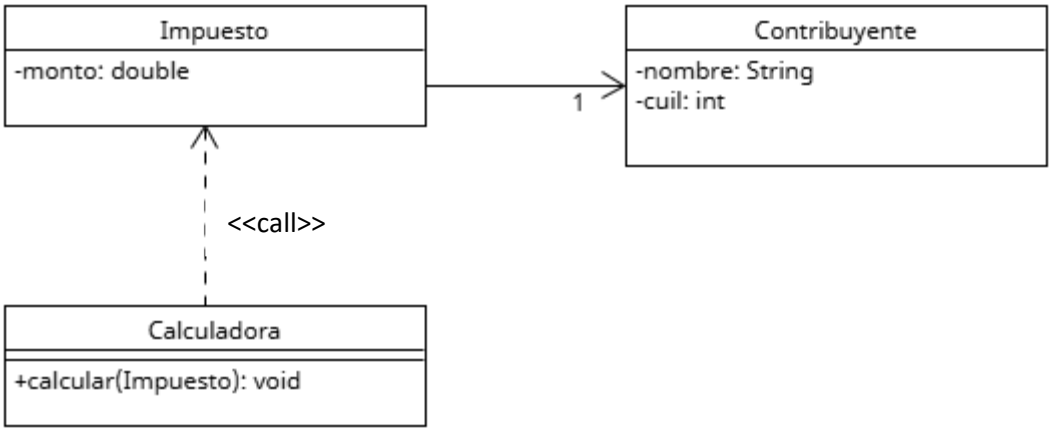
iii. Calculadora->método: void calcular(Impuesto impuesto)

Las clases en Java:

```
public class Calculadora {
    // Dependencia de uso: usa Impuesto en el método
    public void calcular(Impuesto impuesto) {
        System.out.println("Calculando impuesto de $" +
impuesto.getMonto()
```

```
            + " para el contribuyente: " +
impuesto.getContribuyente().getNombre()
            + " (CUIL: " + impuesto.getContribuyente().getCuil() +
")");
    }
}
public class Contribuyente {
    private String nombre;
    private int cuil;

    public String getNombre() {
        return nombre;
    }
    public int getCuil() {
        return cuil;
    }
}
public class Impuesto {
    private double monto;
    private Contribuyente contribuyente; //Asociacion unidireccional
    public double getMonto() {
        return monto;
    }
    public Contribuyente getContribuyente() {
        return contribuyente;
    }
```

}




**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.

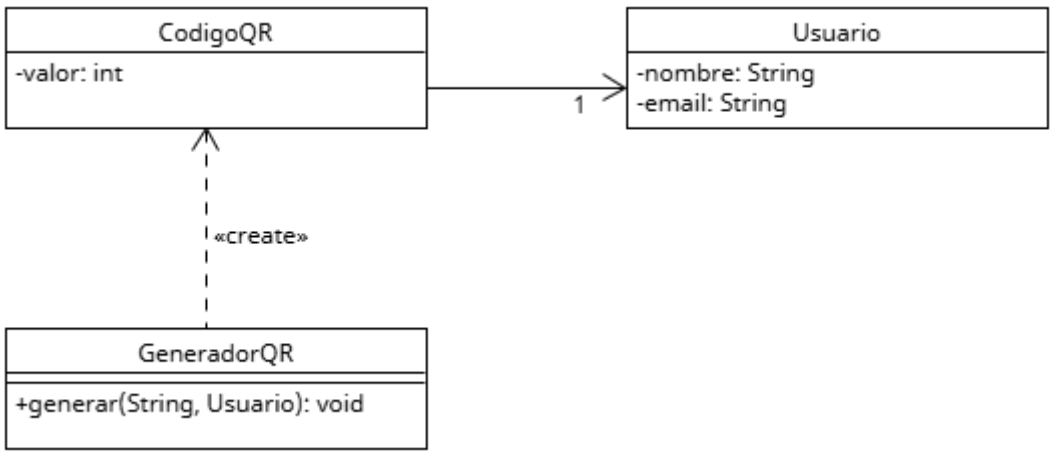
ii. Usuario: nombre, email.

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

Las clases en Java:

```
// Dependencia de creación: genera el objeto dentro del método
    public void generar(int valor, Usuario usuario) {
        CodigoQR qr = new CodigoQR(valor, usuario);
        System.out.println("Código QR generado con valor: " +
qr.getValor() + " para el usuario: " + qr.getUsuario().getNombre() +
                           " (" + qr.getUsuario().getEmail() + ")");
    }
}
public class Usuario {
    private String nombre;
    private String email;
```

```
    public String getNombre() {
        return nombre;
    }
    public String getEmail() {
        return email;
    }
}
public class CodigoQR {
    private int valor;
    private Usuario usuario; //Asociacion unidireccional

    public CodigoQR(int valor, Usuario usuario) {
        this.valor = valor;
        this.usuario = usuario;
    }
    public int getValor() {
        return valor;
    }
    public Usuario getUsuario() {
        return usuario;
    }
}
```



14. EditorVideo - Proyecto - Render

a. Asociación unidireccional: Render → Proyecto

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

c. Clases y atributos:

i. Render: formato.

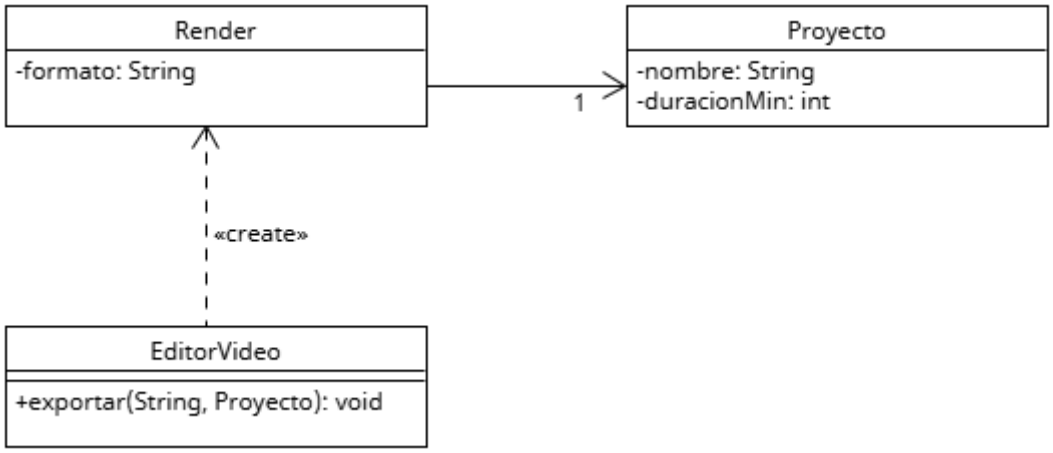
ii. Proyecto: nombre, duracionMin.

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

Las clases en Java:

```
public class EditorVideo {


    public void exportar(String formato, Proyecto proyecto){
        Render render = new Render(formato, proyecto);
        System.out.println("Exportando proyecto '" +
proyecto.getNombre() + "' (" + proyecto.getDuracionMin() + " min) "
                           + "en formato: " + render.getFormato());
    }
}
public class Proyecto {
    private String nombre;
    private int duracionMin;


    public String getNombre() {
        return nombre;
    }
    public int getDuracionMin() {
        return duracionMin;
    }
}
public class Render {
    private String formato;
    private Proyecto proyecto; //Asociacion unidireccional


    public Render(String formato, Proyecto proyecto) {
```

```
        this.formato = formato;
        this.proyecto = proyecto;
    }
    public String getFormato() {
        return formato;
    }
    public Proyecto getProyecto() {
        return proyecto;
    }
}
```




**Repositorio:** https://github.com/Tobias-L7/Trabajos-Practicos-P2.git