Universidad Tecnológica Nacional

TP 5 – Relaciones UML 1 a 1

🔗 Enlace al repositorio de GitHub

Programación II
Belén Yarde Buller
Comisión 3

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

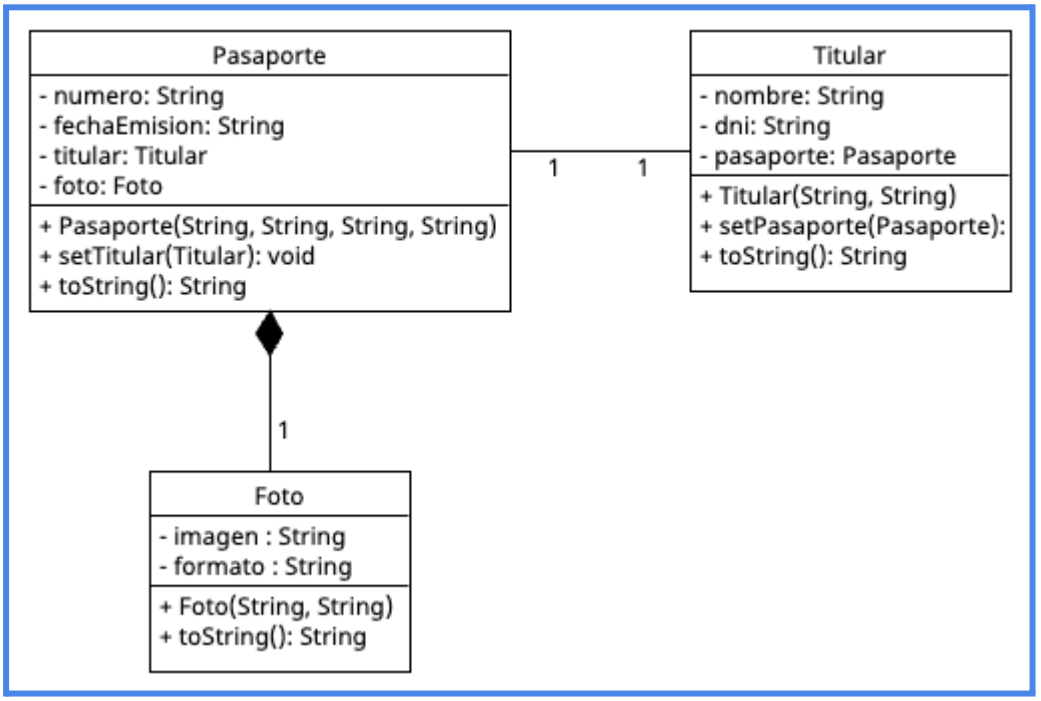
1. Pasaporte - Foto - Titular

a. Composición: Pasaporte → Foto
b. Asociación bidireccional: Pasaporte ↔ Titular

Clases y atributos:

i. Pasaporte: numero, fechaEmision
ii. Foto: imagen, formato
iii. Titular: nombre, dni

Diagrama UML



Tipo de relación y dirección

- ❖ Composición: Pasaporte → Foto
- ❖ Asociación bidireccional: Pasaporte ↔ Titular

Implementación de las clases con atributos y relaciones definidas

```
public class Titular {
    private String nombre;
    private String dni;
    private Pasaporte pasaporte; // Asociación 1..1
```

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

    // Asociación 1..1 bidireccional Pasaporte <-> Titular
    public void setPasaporte(Pasaporte pasaporte) {
        this.pasaporte = pasaporte;
        // Revisa si hay vínculos previos para no entrar en un bucle infinito
        if (pasaporte != null && pasaporte.getTitular() != this) {
            // Establece el vínculo en el otro lado si hace falta
            pasaporte.setTitular(this);
        }
    }

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public Pasaporte getPasaporte() {
        return pasaporte;
    }

    @Override
    public String toString() {
        return """
               ---
               Titular:
               Nombre: """ + nombre
                + "\nDni: " + dni
                + "\nPasaporte: " + (pasaporte != null ? pasaporte.getNumero() : "Sin pasaporte");
    }

}
public class Pasaporte {
    private final String numero;
    private final String fechaEmision;
    private Titular titular; // Asociación 1..1
    private Foto foto; // Composición

    public Pasaporte(String numero, String fechaEmision, String imagen, String formato) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(imagen, formato); // Se crea en el constructor y no hay setter
    }

    // Asociación 1..1 bidireccional Titular <-> Pasaporte
    public void setTitular(Titular titular) {
        this.titular = titular;
        // Revisa si hay vínculos previos para no entrar en un bucle infinito
        if (titular != null && titular.getPasaporte() != this) {
```

```
            // Establece el vínculo en el otro lado si hace falta
            titular.setPasaporte(this);
        }
    }

    public String getNumero() {
        return numero;
    }

    public String getFechaEmision() {
        return fechaEmision;
    }

    public Foto getFoto() {
        return foto;
    }

    public Titular getTitular() {
        return titular;
    }

    @Override
    public String toString() {
        return """
               ---
               Pasaporte:
               Numero: """ + numero
                + "\nFecha de emisión: " + fechaEmision
                + "\nTitular: " + (titular != null ? titular.getNombre() : "Sin titular");
    }

}
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;
    }

    @Override
    public String toString() {
        return "Foto: " + imagen + "." + formato;
    }

}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 1
        Titular titular = new Titular("Belén Yarde Buller", "37123456");
        Pasaporte pasaporte = new Pasaporte("A123456", "2023-09-09T00:00:00.000Z", "urlimagen",
"jpg");
```

```
        System.out.println(pasaporte);
        System.out.println(titular);
        // Asociamos Titular <-> Pasaporte con un solo un setter:
        titular.setPasaporte(pasaporte);
        System.out.println(pasaporte);
        System.out.println(titular);
        System.out.println(pasaporte.getFoto());
    }
}
```

Ejecución del código en consola:

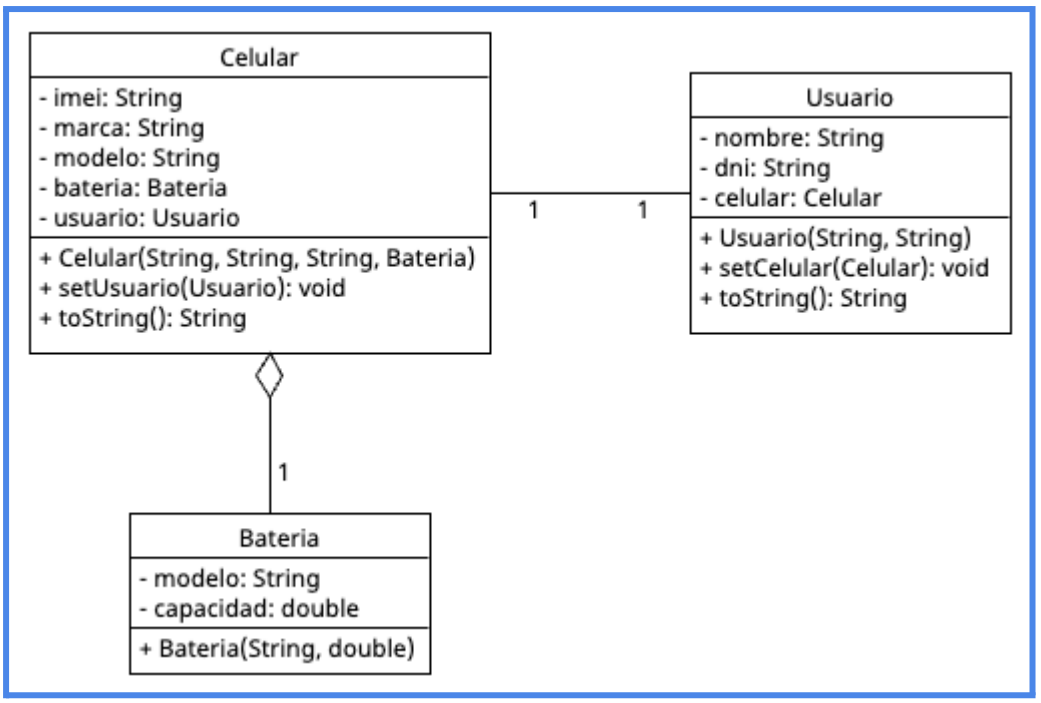
```
---
Pasaporte:
Numero:A123456
Fecha de emisión: 2023-09-09T00:00:00.000Z
Titular: Sin titular
---
Titular:
Nombre:Belén Yarde Buller
Dni: 37123456
Pasaporte: Sin pasaporte
---
Pasaporte:
Numero:A123456
Fecha de emisión: 2023-09-09T00:00:00.000Z
Titular: Belén Yarde Buller
---
Titular:
Nombre:Belén Yarde Buller
Dni: 37123456
Pasaporte: A123456
Foto: urlimagen.jpg
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Agregación: Celular → Batería
- Asociación bidireccional: Celular ↔ Usuario

Implementación de las clases con atributos y relaciones definidas

```
public class Usuario {
    private final String nombre;
    private final String dni;
    private Celular celular; // Asociación bidireccional 1..1

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

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

    public Celular getCelular() {
        return celular;
    }

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

    public void setNombre(String nombre) {
```

```java
        this.nombre = nombre;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    @Override
    public String toString() {
        String celInfo = (celular != null)
                ? celular.getMarca() + " " + celular.getModelo() + " [" + celular.getImei() + "]"
                : "sin celular";
        return "Usuario{"
                + "nombre='" + nombre + '\''
                + ", dni='" + dni + '\''
                + ", celular=" + celInfo
                + '}';
    }

}
public class Celular {
    private String imei;
    private String marca;
    private String modelo;
    // Agregación: Una clase está compuesta por otra, pero esta puede existir por separado:
    private Bateria bateria;
    private Usuario usuario; // Asociación bidireccional 1..1

    // Agregación: Se pasa la instancia como parámetro en el constructor.
    public Celular(String imei, String marca, String modelo, Bateria bateria) {
        this.imei = imei;
        this.marca = marca;
        this.modelo = modelo;
        this.bateria = bateria;
    }

    public String getImei() {
        return imei;
    }

    public String getMarca() {
        return marca;
    }

    public String getModelo() {
        return modelo;
    }

    public Bateria getBateria() {
        return bateria;
    }

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
        if (usuario != null && usuario.getCelular() != this) {
            usuario.setCelular(this);
        }
    }
```

```java
    public void setImei(String imei) {
        this.imei = imei;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

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

    @Override
    public String toString() {
        String usuarioInfo = (usuario != null)
                ? usuario.getNombre() + " (" + usuario.getDni() + ")"
                : "sin usuario";
        return "Celular{"
                + "imei='" + imei + '\''
                + ", marca='" + marca + '\''
                + ", modelo='" + modelo + '\''
                + ", bateria=" + bateria
                + // OK: Bateria no referencia a Celular/Usuario
                ", usuario=" + usuarioInfo
                + '}';
    }

}

public class Bateria {
    private String modelo;
    private double capacidad;

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

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public double getCapacidad() {
        return capacidad;
    }

    public void setCapacidad(double capacidad) {
        this.capacidad = capacidad;
    }
}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 2
        Bateria bateria = new Bateria("Samsung-BX100", 4500);
```

```
        Celular celular = new Celular("123456789012345", "Samsung", "Galaxy S24", bateria);
        Usuario usuario = new Usuario("Ana López", "40111222");
        // Basta con un solo setter para establecer la relación
        usuario.setCelular(celular);
        System.out.println(usuario.toString());
        System.out.println(celular.toString());
    }
}
```

Ejecución del código en consola:

```
run:
Usuario{nombre='Ana López', dni='40111222', celular=Samsung Galaxy S24 [123456789012345]}
Celular{imei='123456789012345', marca='Samsung', modelo='Galaxy S24', bateria=tp5.Bateria@452b3a41,
usuario=Ana López (40111222)}
BUILD SUCCESSFUL (total time: 0 seconds)
```

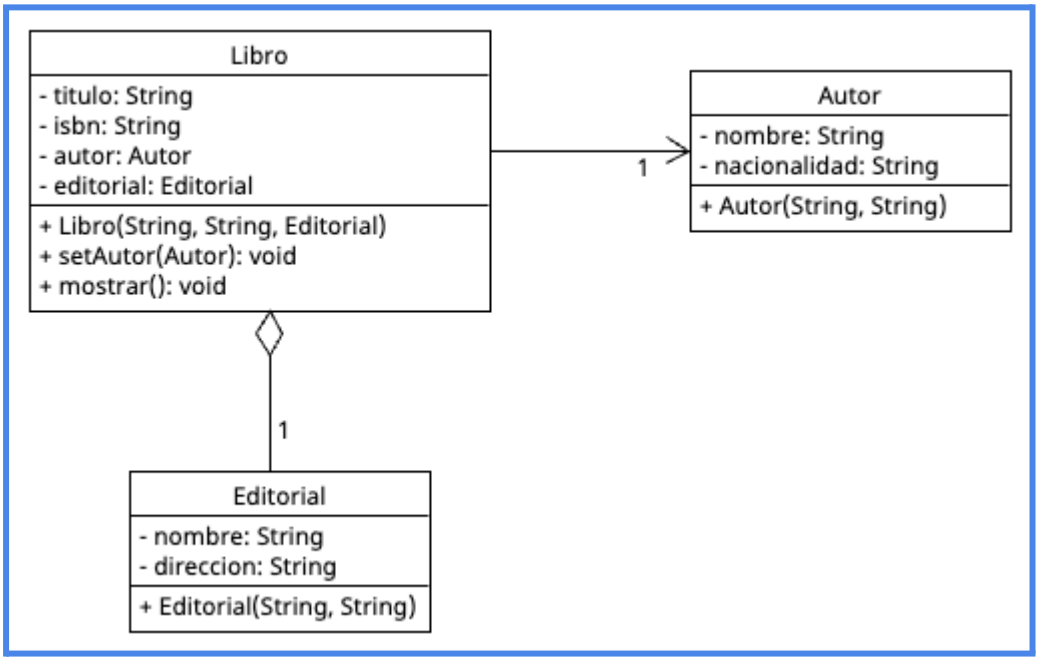
3. Libro - Autor - Editorial

a. Asociación unidireccional: Libro → Autor
b. Agregación: Libro → Editorial

Clases y atributos:

i. Libro: titulo, isbn
ii. Autor: nombre, nacionalidad
iii. Editorial: nombre, direccion

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: Libro → Autor

- ❖ Agregación: Libro → Editorial

Implementación de las clases con atributos y relaciones definidas

```
public class Libro {
    private String titulo;
    private String isbn;
    private Autor autor; // Asociación unidireccional Libro -> Autor
    private Editorial editorial; // Agregación

    // Agregación - Implementación:
    // Se pasa la instancia como parámetro en el constructor
    public Libro(String titulo, String isbn, Editorial editorial) {
        this.titulo = titulo;
        this.isbn = isbn;
        this.editorial = editorial;
    }

     public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public Editorial getEditorial() {
        return editorial;
    }

    public void setEditorial(Editorial editorial) {
        this.editorial = editorial;
    }

    public void setAutor(Autor autor) {
        this.autor = autor;
    }

    public Autor getAutor() {
        return autor;
    }

    public void mostrar() {
        System.out.println("Libro: " + titulo + " - Autor: " + autor.getNombre() + " - Editorial: "
+ editorial.getNombre());
    }
}

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;
    }

    public void setNacionalidad(String nacionalidad) {
        this.nacionalidad = nacionalidad;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

}

public class Editorial {
    private String nombre;
    private String direccion;

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

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDireccion() {
        return direccion;
    }

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

public class Main {
    public static void main(String[] args) {
        // Ejercicio 3
        Autor autor = new Autor("Jorge Luis Borges", "Argentina");
        Editorial editorial = new Editorial("Sudamericana", "Humberto Primo 555, Capital Federal");
        Libro libro = new Libro("El hacedor", "950-04-0163-0", editorial);
        libro.setAutor(autor);
        libro.mostrar();
    }
}
```

Ejecución del código en consola:

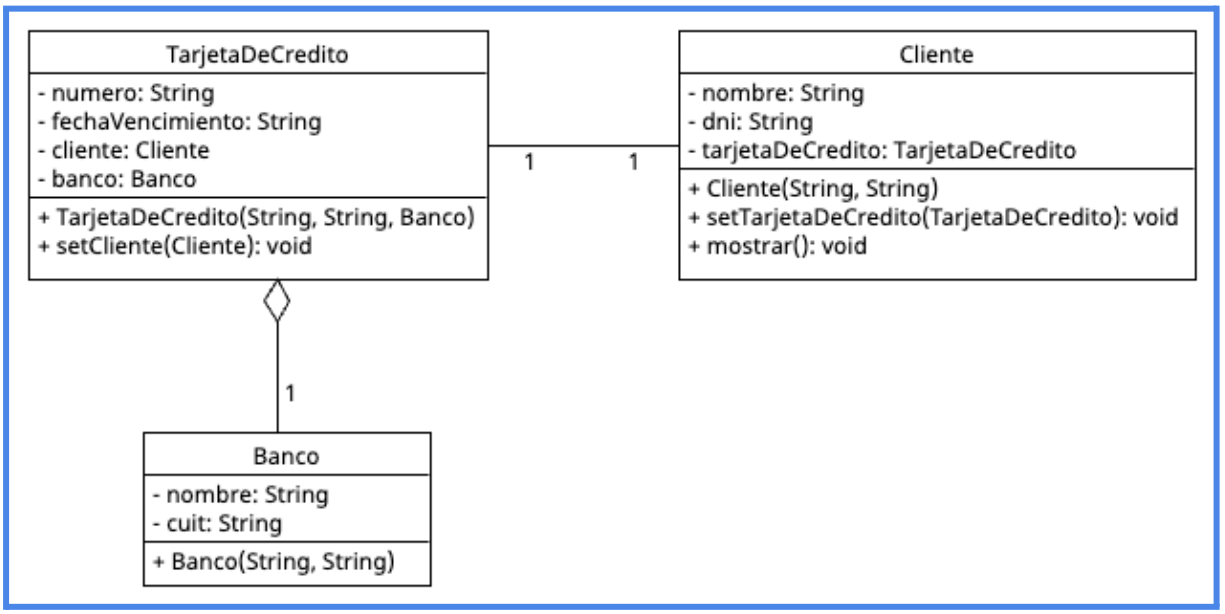
```
run:
Libro: El hacedor - Autor: Jorge Luis Borges - Editorial: Sudamericana
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Asociación bidireccional: TarjetaDeCrédito ↔ Cliente
- Agregación: TarjetaDeCrédito → Banco

Implementación de las clases con atributos y relaciones definidas

```
public class TarjetaDeCredito {
    private String numero;
    private String fechaVencimiento;
    private Cliente cliente; // Asociación 1..1 bidireccional
    private Banco banco; // Agregación

    public TarjetaDeCredito(String numero, String fechaVencimiento, Banco banco) {
        this.numero = numero;
        this.fechaVencimiento = fechaVencimiento;
        this.banco = banco;
    }

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

```java
    public String getFechaVencimiento() {
        return fechaVencimiento;
    }

    public Cliente getCliente() {
        return cliente;
    }

    public Banco getBanco() {
        return banco;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public void setFechaVencimiento(String fechaVencimiento) {
        this.fechaVencimiento = fechaVencimiento;
    }

    public void setBanco(Banco banco) {
        this.banco = banco;
    }

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

public class Cliente {
    private String nombre;
    private String dni;
    private TarjetaDeCredito tarjetaDeCredito;  // Asociaciòn 1..1 bidireccional

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

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

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

    public TarjetaDeCredito getTarjetaDeCredito() {
        return tarjetaDeCredito;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
```

```
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public void mostrar() {
        System.out.println("Cliente: " + nombre);
        System.out.println("Tarjeta: " + tarjetaDeCredito.getNumero());
        System.out.println("Banco: " + tarjetaDeCredito.getBanco().getNombre());
    }
}

public class Banco {
    private String nombre;
    private String cuit;

    public Banco(String nombre, String cuit) {
        this.nombre = nombre;
        this.cuit = cuit;
    }

    public String getNombre() {
        return nombre;
    }

    public String getCuit() {
        return cuit;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setCuit(String cuit) {
        this.cuit = cuit;
    }
}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 4
        Cliente cliente = new Cliente("Belén Yarde Buller", "37000000");
        Banco banco = new Banco("Galicia", "30-50000173-5");
        TarjetaDeCredito tarjetaDeCredito = new TarjetaDeCredito("4517698809093472",
"2023-09-09T00:00:00.000Z", banco);
        cliente.setTarjetaDeCredito(tarjetaDeCredito); // Solo un setter necesario
        cliente.mostrar();
    }
}
```

Ejecución del código en consola:

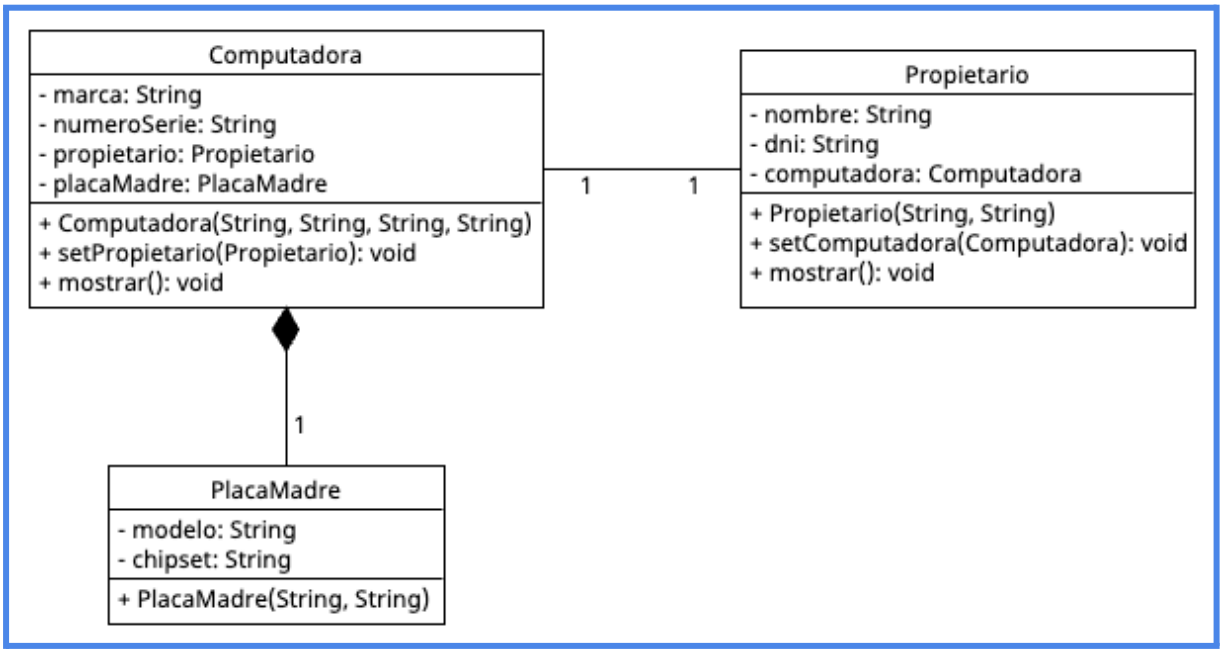
```
run:
Cliente: Belén Yarde Buller
Tarjeta: 4517698809093472
Banco: Galicia
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Composición: Computadora → PlacaMadre
- Asociación bidireccional: Computadora ↔ Propietario

Implementación de las clases con atributos y relaciones definidas

```
public class Computadora {
    private String marca;
    private String numeroSerie;
    private Propietario propietario; // Asociación bidireccional 1..1
    private PlacaMadre placaMadre; // Composición

    public Computadora(String marca, String numeroSerie, String modelo, String chipset) {
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        // Se crea internamente la instancia a través de parámetros
        // primitivos del constructor:
        this.placaMadre = new PlacaMadre(modelo, chipset);
    }

    public Propietario getPropietario() {
        return propietario;
    }

    public String getMarca() {
```

```
        return marca;
    }

    public String getNumeroSerie() {
        return numeroSerie;
    }

    public void setPropietario(Propietario propietario) {
        this.propietario = propietario;
        if (propietario != null && propietario.getComputadora() != this) {
            propietario.setComputadora(this);
        }
    }

    public void mostrar() {
        System.out.println("Computadora:"
                + "\n- Marca: " + marca
                + "\n- Número de serie: " + numeroSerie
                + "\n- Placa madre: " + placaMadre.getModelo()
                + "\n- Propietario: " + propietario.getNombre() + ", DNI: "
                + propietario.getDni());
    }

}

public class Propietario {
    private String nombre;
    private String dni;
    private Computadora computadora; // Asociación bidireccional 1..1

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

    public Computadora getComputadora() {
        return computadora;
    }

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

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

    public void mostrar() {
        System.out.println("Propietario:"
                + "\n- nombre: " + nombre
                + "\n- computadora: " + computadora.getMarca() + " "
                + computadora.getNumeroSerie());
    }
}

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 class Main {
    public static void main(String[] args) {
        // Ejercicio 5
        Propietario propietario = new Propietario("Belén", "37000000");
        // La clase Placa Madre no se crea en el Main sino directamente desde el constructor
        // de la clase Computadora
        Computadora computadora = new Computadora("HP", "HP123-X321", "Micro ATX", "Intel b760");
        computadora.setPropietario(propietario); // Solo un setter necesario
        computadora.mostrar();
    }
}
```

Ejecución del código en consola:

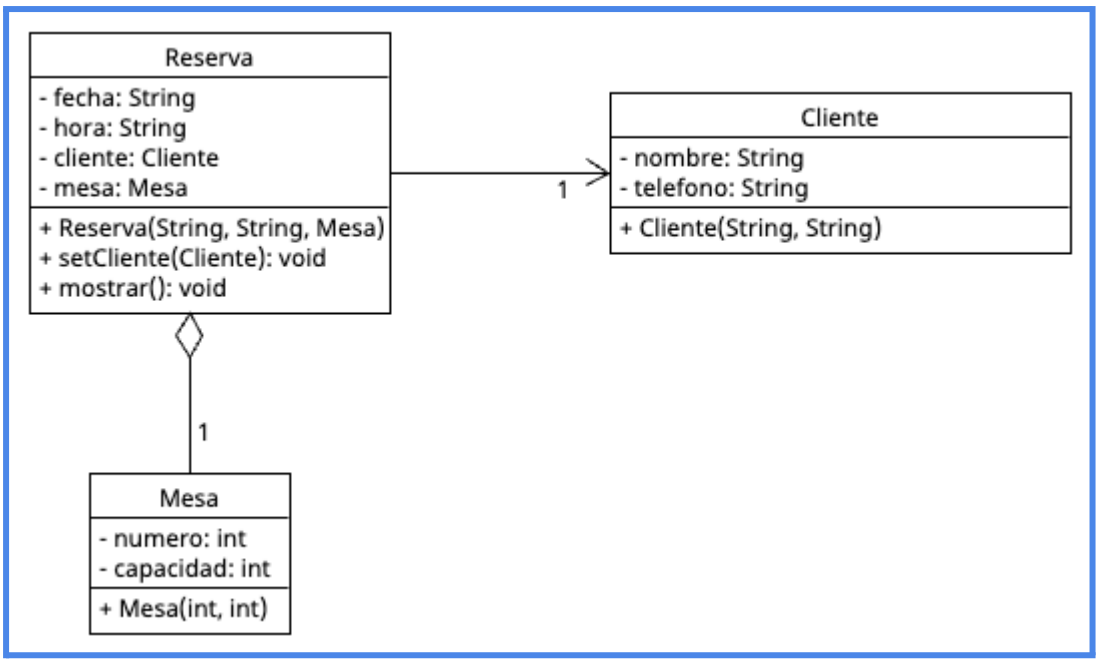
```
run:
Computadora:
- Marca: HP
- Número de serie: HP123-X321
- Placa madre: Micro ATX
- Propietario: Belén, DNI: 37000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: Reserva → Cliente
- Agregación: Reserva → Mesa

Implementación de las clases con atributos y relaciones definidas

```
public class Reserva {
    private String fecha;
    private String hora;
    private Cliente cliente; // Asociación unidireccional Reserva -> Cliente
    private Mesa mesa; // Agregación

    // Agregación: se pasa la instancia como parámtro en el constructor
    public Reserva(String fecha, String hora, Mesa mesa) {
        this.fecha = fecha;
        this.hora = hora;
        this.mesa = mesa;
    }

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    public void mostrar() {
        System.out.println("Reserva: "
                + "\n- Fecha: " + fecha
                + "\n- Hora: " + hora
                + "\n- Cliente: " + cliente.getNombre()
                + "\n- Mesa nro: " + mesa.getNumero()
        );
    }
}

public class Cliente {
    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;
    }

}

public class Mesa {
    private int numero;
    private int capacidad;

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

    public int getNumero() {
        return numero;
    }

    public int getCapacidad() {
        return capacidad;
    }
}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 6
        Mesa mesa = new Mesa(5, 10);
        Reserva reserva = new Reserva("2023-09-09", "19:00:00", mesa);
        Cliente cliente = new Cliente("Belén", "1590902929");
        reserva.setCliente(cliente);
        reserva.mostrar();
    }
}
```

Ejecución del código en consola:

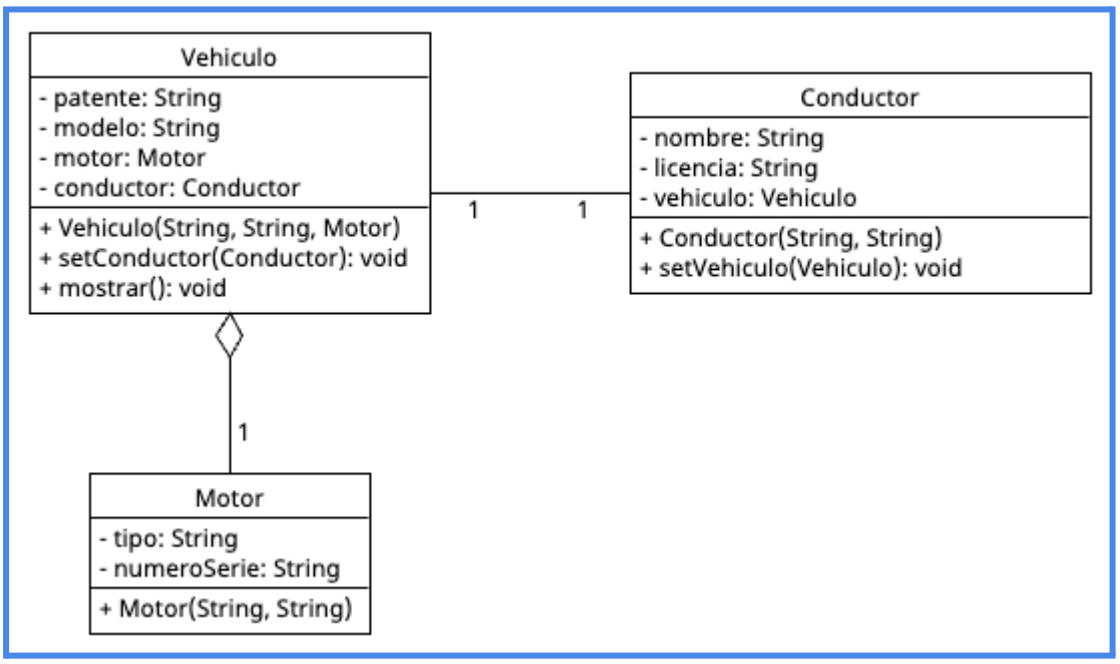
```
run:
Reserva:
- Fecha: 2023-09-09
- Hora: 19:00:00
- Cliente: Belén
- Mesa nro: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Agregación: Vehículo → Motor
- Asociación bidireccional: Vehículo ↔ Conductor

Implementación de las clases con atributos y relaciones definidas

```
public class Vehiculo {
    private String patente;
    private String modelo;
    private Motor motor; // Agregación Vehículo -> Motor
    private Conductor conductor; // Asociación bidireccional 1..1

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

    public Conductor getConductor() {
        return conductor;
    }

    public void setConductor(Conductor conductor) {
        this.conductor = conductor;
        if (conductor != null && conductor.getVehiculo() != this) {
            conductor.setVehiculo(this);
        }
    }
```

```
    public void mostrar() {
        System.out.println("Vehículo: " + modelo +
                "\nMotor: " + motor.getTipo() +
                "\nConductor: " + conductor.getNombre() +
                ", licencia: " + conductor.getLicencia());
    }
}

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

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

    public String getTipo() {
        return tipo;
    }

    public String getNumeroSerie() {
        return numeroSerie;
    }


}

public class Conductor {
    private String nombre;
    private String licencia;
    private Vehiculo vehiculo;

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

    public String getNombre() {
        return nombre;
    }

    public String getLicencia() {
        return licencia;
    }

    public Vehiculo getVehiculo() {
        return vehiculo;
    }

    public void setVehiculo(Vehiculo vehiculo) {
        this.vehiculo = vehiculo;
        if (vehiculo != null && vehiculo.getConductor() != this){
            vehiculo.setConductor(this);
        }
    }

}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 7
        Motor motor = new Motor("Hibrido", "H123-345F");
```

```
        Vehiculo vehiculo = new Vehiculo("ABC123", "Focus", motor);
        Conductor conductor = new Conductor("Belén", "LIC123");
        vehiculo.setConductor(conductor);
        vehiculo.mostrar();
    }
}
```

Ejecución del código en consola:

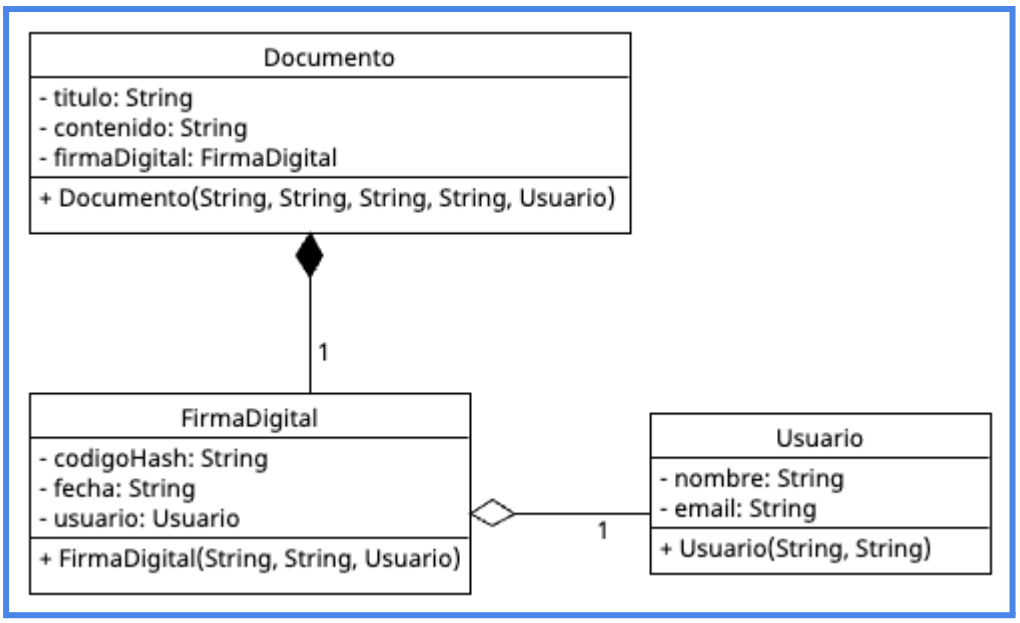
```
run:
Vehículo: Focus
Motor: Hibrido
Conductor: Belén, licencia: LIC123
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Composición: Documento → FirmaDigital
- Agregación: FirmaDigital → Usuario

Implementación de las clases con atributos y relaciones definidas

```java
public class Documento {
    private String titulo;
    private String contenido;
    private FirmaDigital firmaDigital; // Composición Documento -> FirmaDigital

    // Composición: Se crea internamente la instancia a través de parámetros
    // primitivos del constructor:
    public Documento(String titulo, String contenido, String codigoHash, String fecha, Usuario
usuario) {
        this.titulo = titulo;
        this.contenido = contenido;
        this.firmaDigital = new FirmaDigital(codigoHash, fecha, usuario);
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getContenido() {
        return contenido;
    }

    public void setContenido(String contenido) {
        this.contenido = contenido;
    }

    public FirmaDigital getFirmaDigital() {
        return firmaDigital;
    }

    public void setFirmaDigital(FirmaDigital firmaDigital) {
        this.firmaDigital = firmaDigital;
    }
}

public class FirmaDigital {
    private String codigoHash;
    private String fecha;
    private Usuario usuario; // Agregación FirmaDigital -> Usuario

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

    public String getCodigoHash() {
        return codigoHash;
    }

    public void setCodigoHash(String codigoHash) {
        this.codigoHash = codigoHash;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
```

```
        this.fecha = fecha;
    }

    public Usuario getUsuario() {
        return usuario;
    }

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

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;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
public class Main {
    public static void main(String[] args) {
        // Ejercicio 8
        Usuario usuario = new Usuario("belenyb", "belen@mail.com");
        Documento doc = new Documento("Título documento", "Contenido del documento.", "xqwe2189f",
"2025-08-01", usuario);
        System.out.println("Documento: " + doc.getTitulo());
        System.out.println("Usuario de la firma: " +
doc.getFirmaDigital().getUsuario().getNombre());
        System.out.println("Hash de firma: " + doc.getFirmaDigital().getCodigoHash());
    }
}
```

Ejecución del código en consola:

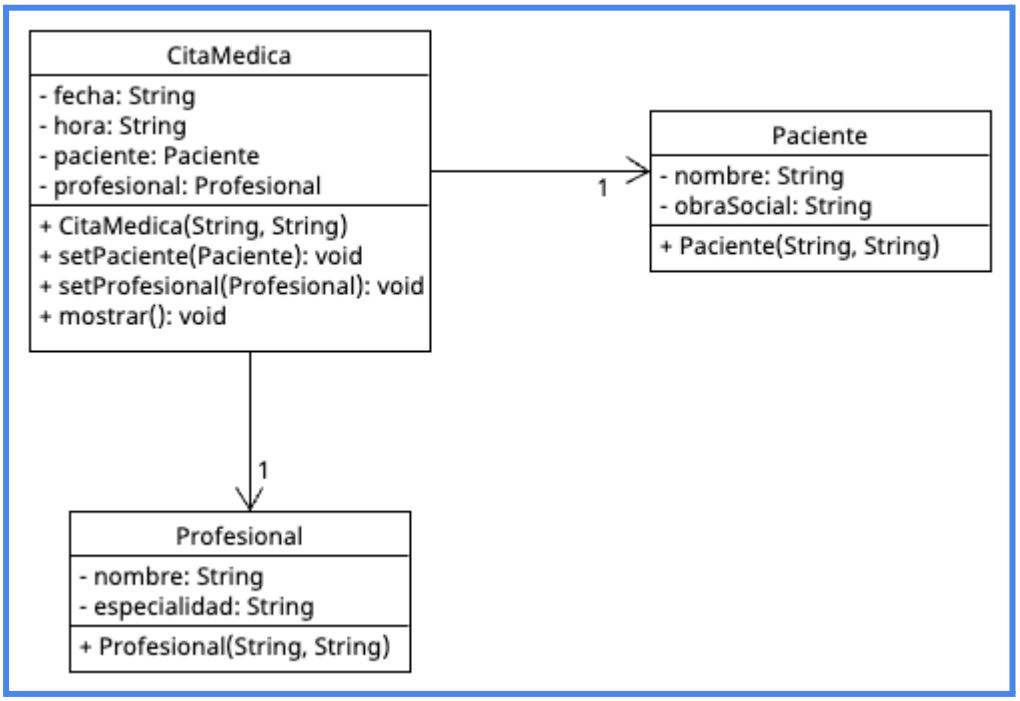
```
run:
Documento: Título documento
Usuario de la firma: belenyb
Hash de firma: xqwe2189f
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: CitaMedica → Paciente
- Asociación unidirecciona: CitaMedica → Profesional

Implementación de las clases con atributos y relaciones definidas

```java
public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente; // Asociación unidireccional CitaMedica -> Paciente
    private Profesional profesional; // Asociación unidireccional CitaMedica -> Profesional

    public CitaMedica(String fecha, String hora) {
        this.fecha = fecha;
        this.hora = hora;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }
```

```java
    public String getHora() {
        return hora;
    }

    public void setHora(String hora) {
        this.hora = hora;
    }

    public Paciente getPaciente() {
        return paciente;
    }

    public void setPaciente(Paciente paciente) {
        this.paciente = paciente;
    }

    public Profesional getProfesional() {
        return profesional;
    }

    public void setProfesional(Profesional profesional) {
        this.profesional = profesional;
    }

    public void mostrar() {
        System.out.println("Cita médica de: "
                + paciente.getNombre()
                + " (" + paciente.getObraSocial() + ")"
                + "\nFecha y hora: " + fecha + " " + hora
                + "\n Profesional: " + profesional.getNombre()
                + " (" + profesional.getEspecialidad() + ")"
        );
    }

}

public class Paciente {
    private String nombre;
    private String obraSocial;

    public Paciente(String nombre, String obraSocial) {
        this.nombre = nombre;
        this.obraSocial = obraSocial;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getObraSocial() {
        return obraSocial;
    }

    public void setObraSocial(String obraSocial) {
        this.obraSocial = obraSocial;
    }

}
```

```java
public class Profesional {
    private String nombre;
    private String especialidad;

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

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEspecialidad() {
        return especialidad;
    }

    public void setEspecialidad(String especialidad) {
        this.especialidad = especialidad;
    }


}

public class Main {
        // Ejercicio 9
        Paciente paciente = new Paciente("Belen", "Medicus");
        Profesional profesional = new Profesional("Mariana", "Hepatología");
        CitaMedica citaMedica = new CitaMedica("2023-09-09", "19:00:00");
        citaMedica.setPaciente(paciente);
        citaMedica.setProfesional(profesional);
        citaMedica.mostrar();
    }
}
```

Ejecución del código en consola:

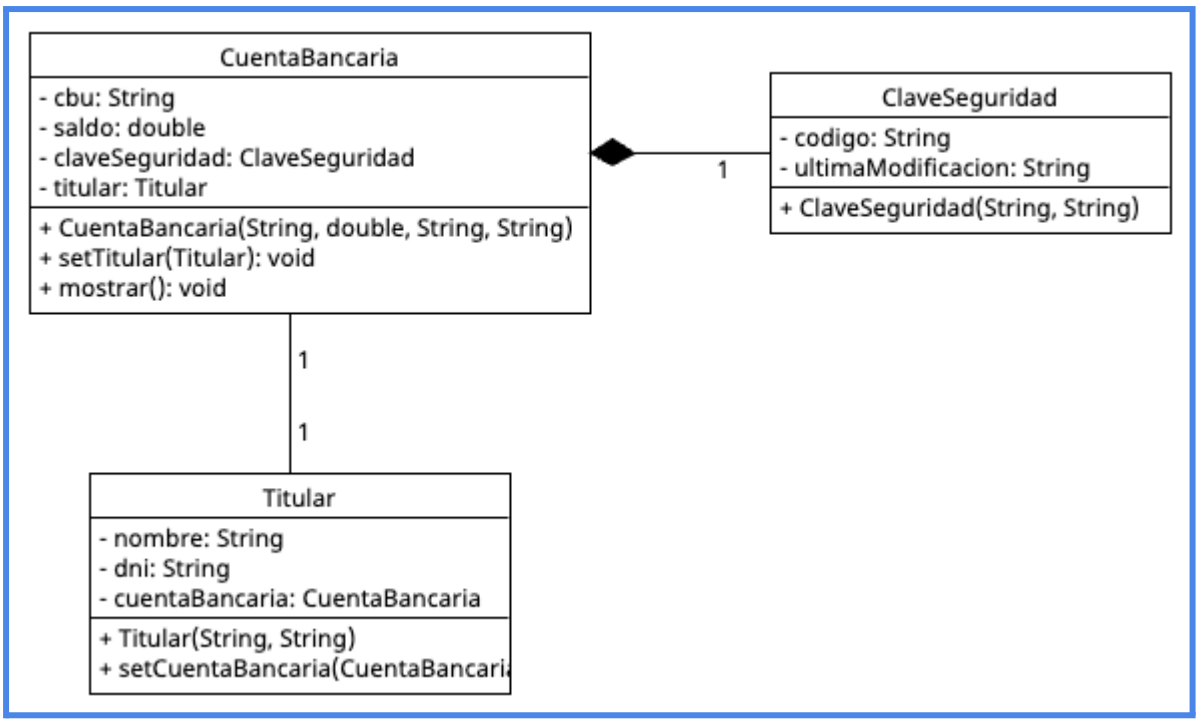
```
Cita médica de: Belen (Medicus)
Fecha y hora: 2023-09-09 19:00:00
 Profesional: Mariana (Hepatología)
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Diagrama UML



Tipo de relación y dirección

- Composición: CuentaBancaria → ClaveSeguridad
- Asociación bidireccional: CuentaBancaria ↔ Titular

Implementación de las clases con atributos y relaciones definidas

```
public class CuentaBancaria {
    private String cbu;
    private double saldo;
    private ClaveSeguridad claveSeguridad; // Composición CuentaBancaria -> Clave Seguridad
    private Titular titular; // Asociación bidireccional 1..1

    // Composición: Se crea internamente la instancia a través de parámetros
    // primitivos del constructor:
    public CuentaBancaria(String cbu, double saldo, String codigo, String ultimaModificacion) {
        this.cbu = cbu;
        this.saldo = saldo;
        this.claveSeguridad = new ClaveSeguridad(codigo, ultimaModificacion);
    }

    public String getCbu() {
        return cbu;
    }

    public void setCbu(String cbu) {
        this.cbu = cbu;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }
```

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

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

    public void mostrar() {
        System.out.println(
                "Cuenta Bancaria:"
                + "\nCBU: " + cbu
                + "\nSaldo: " + saldo
                + "\nClave de seguridad: " + claveSeguridad.getCodigo()
                + "\nTitular: " + titular.getNombre()
        );
    }

}

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 void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getUltimaModificacion() {
        return ultimaModificacion;
    }

    public void setUltimaModificacion(String ultimaModificacion) {
        this.ultimaModificacion = ultimaModificacion;
    }

}

public class Titular {
    private String nombre;
    private String dni;
    private CuentaBancaria cuentaBancaria; // Asociación bidireccional 1..1

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

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

```
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public CuentaBancaria getCuentaBancaria() {
        return cuentaBancaria;
    }

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

}

public class Main {
    public static void main(String[] args) {
        // Ejercicio 10
        Titular titular = new Titular("Belén Yarde Buller", "37000000");
        CuentaBancaria cuentaBancaria = new CuentaBancaria("27-37000000-1", 340000.45, "Codigo123",
"2023-09-09");
        cuentaBancaria.setTitular(titular);
        cuentaBancaria.mostrar();
    }
}
```

Ejecución del código en consola:

```
Cuenta Bancaria:
CBU: 27-37000000-1
Saldo: 340000.45
Clave de seguridad: Codigo123
Titular: Belén Yarde Buller
BUILD SUCCESSFUL (total time: 0 seconds)
```

DEPENDENCIA DE USO

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

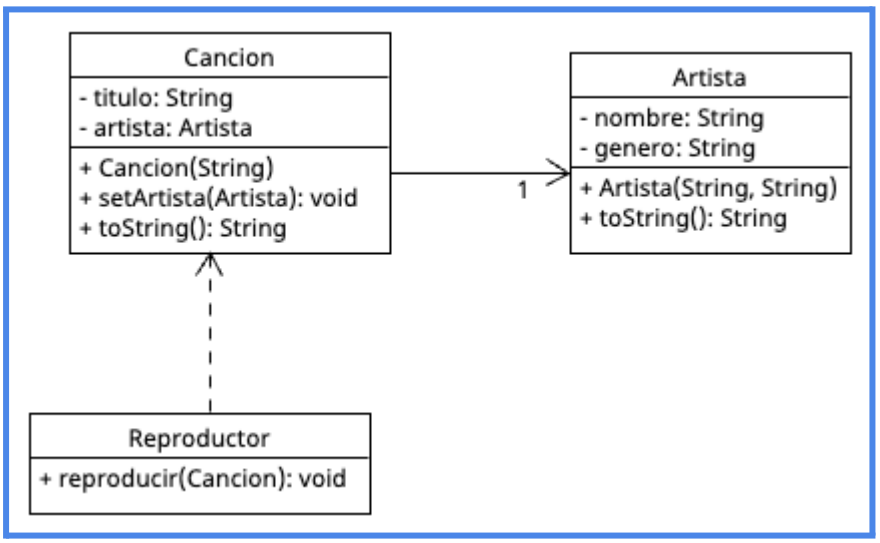
Ejercicios de Dependencia de Uso

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)

Diagrama UML



Tipo de relación y dirección

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

Implementación de las clases con atributos y relaciones definidas

```java
public class Cancion {
    private String titulo;
    private Artista artista; // Asociación unidireccional Cancion -> Artista

    public Cancion(String titulo) {
        this.titulo = titulo;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public Artista getArtista() {
        return artista;
    }

    public void setArtista(Artista artista) {
        this.artista = artista;
    }

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

```
    }

}

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 void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getGenero() {
        return genero;
    }

    public void setGenero(String genero) {
        this.genero = genero;
    }

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

}

public class Reproductor {
    // Dependencia de uso
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciendo: " + cancion.getTitulo() + " (" +
cancion.getArtista().getNombre() + ") " + " 🎶🎶🎶🎶🎶");
    }

}

public class Main {
    public static void main(String[] args) {
        Artista artista = new Artista("Subalterna", "Rock alternativo");
        Cancion cancion = new Cancion("Sobre los paisajes grises");
        cancion.setArtista(artista);
        Reproductor reproductor = new Reproductor();
        reproductor.reproducir(cancion);
    }
}
```

Ejecución del código en consola:

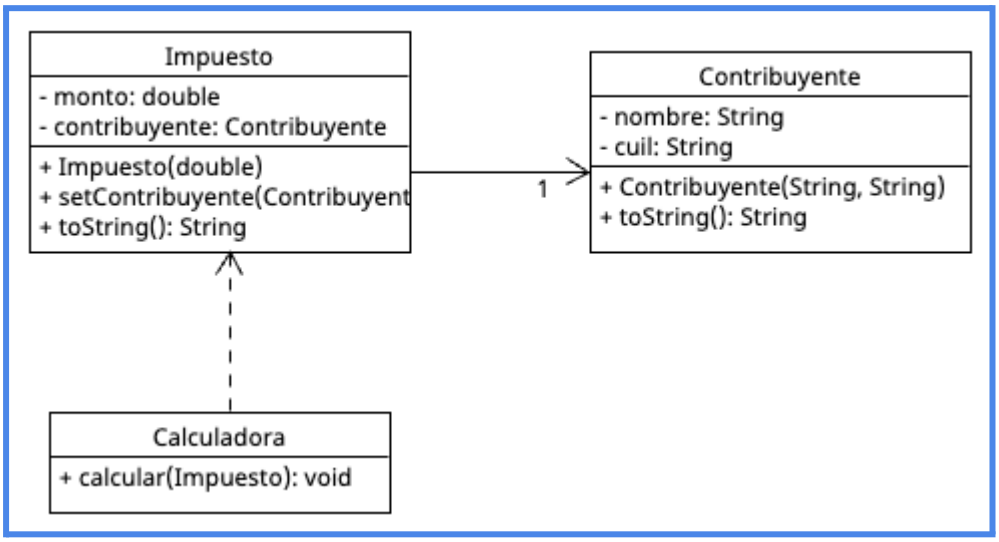
```
run:
Reproduciendo: Sobre los paisajes grises (Subalterna)  🎶🎶🎶🎶🎶
BUILD SUCCESSFUL (total time: 0 seconds)
```

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)

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: Impuesto → Contribuyente
- Dependencia de uso: Calculadora.calcular(Impuesto)

Implementación de las clases con atributos y relaciones definidas

```java
public class Impuesto {
    private double monto;
    private Contribuyente contribuyente;

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

    public double getMonto() {
        return monto;
    }

    public void setMonto(double monto) {
        this.monto = monto;
    }

    public Contribuyente getContribuyente() {
        return contribuyente;
    }
```

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

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

}

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 void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getCuil() {
        return cuil;
    }

    public void setCuil(String cuil) {
        this.cuil = cuil;
    }

    @Override
    public String toString() {
        return "Contribuyente{" + "nombre=" + nombre + ", cuil=" + cuil + '}';
    }

}

public class Calculadora {
    public void calcular(Impuesto impuesto) {
        final int montoEjemplo = 500000;
        double total = montoEjemplo + montoEjemplo * impuesto.getMonto();
        System.out.println("Impuesto aplicado a " + impuesto.getContribuyente().getNombre() + ": " +
impuesto.getMonto() + "\nTotal: " + total);
    }
}

public class Main {
    public static void main(String[] args) {
        Contribuyente contribuyente = new Contribuyente("Belén Yarde Buller", "27-37000000-1");
        Impuesto impuesto = new Impuesto(0.21);
        impuesto.setContribuyente(contribuyente);
        Calculadora calculadora = new Calculadora();
        calculadora.calcular(impuesto); // dependencia de uso (método recibe Impuesto)
    }
}
```

Ejecución del código en consola:

```
run:
Impuesto aplicado a Belén Yarde Buller: 0.21
Total: 605000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

DEPENDENCIA DE CREACIÓN

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

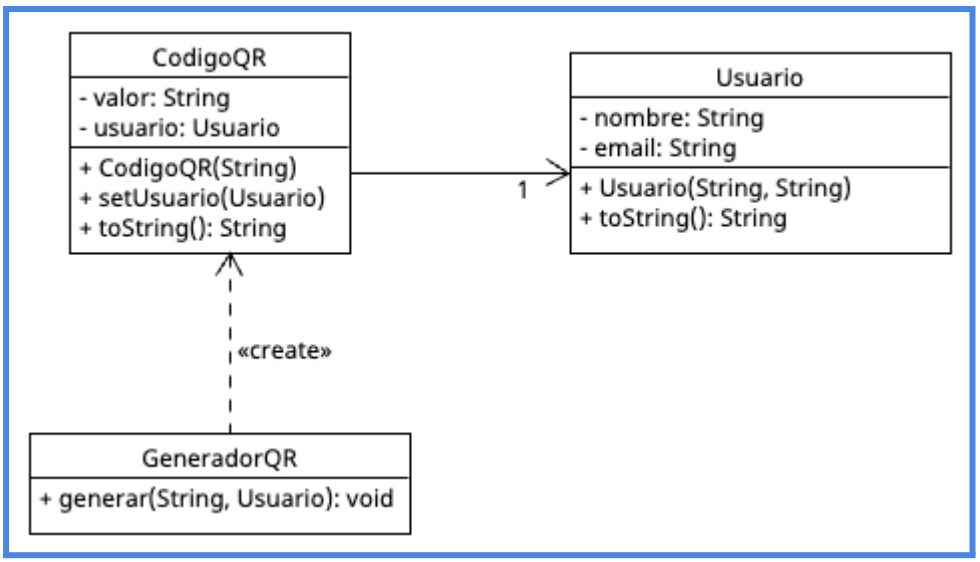
Ejercicios de Dependencia de Creación

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)

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: CódigoQR → Usuario
- Dependencia de creación: GeneradorQR.generar(String, Usuario)

Implementación de las clases con atributos y relaciones definidas

```
public class CodigoQR {
    private String valor;
    private Usuario usuario; // Asociación unidireccional CodigoQR -> Usuario

    public CodigoQR(String valor) {
        this.valor = valor;
```

```java
    }

    public String getValor() {
        return valor;
    }

    public void setValor(String valor) {
        this.valor = valor;
    }

    public Usuario getUsuario() {
        return usuario;
    }

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

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

}

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 void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

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

public class GeneradorQR {
    // Dependencia de Creación: crea un CodigoQR en el método y no lo guarda como atributo.
    public void generar(String valor, Usuario usuario) {
        CodigoQR qr = new CodigoQR(valor);
        qr.setUsuario(usuario); // Asociación unidireccional CodigoQR -> Usuario
        System.out.println("QR generado exitosamente para usuario: " + qr.getUsuario().getNombre());
    }
```

```
}

public class Main {
    public static void main(String[] args) {
        Usuario usuario = new Usuario("belenyb", "belen@mail.com");
        GeneradorQR generadorQr = new GeneradorQR();
        generadorQr.generar("Contenido del código QR", usuario); // Crea instancia de CodigoQR
    }
}
```

Ejecución de código en consola:

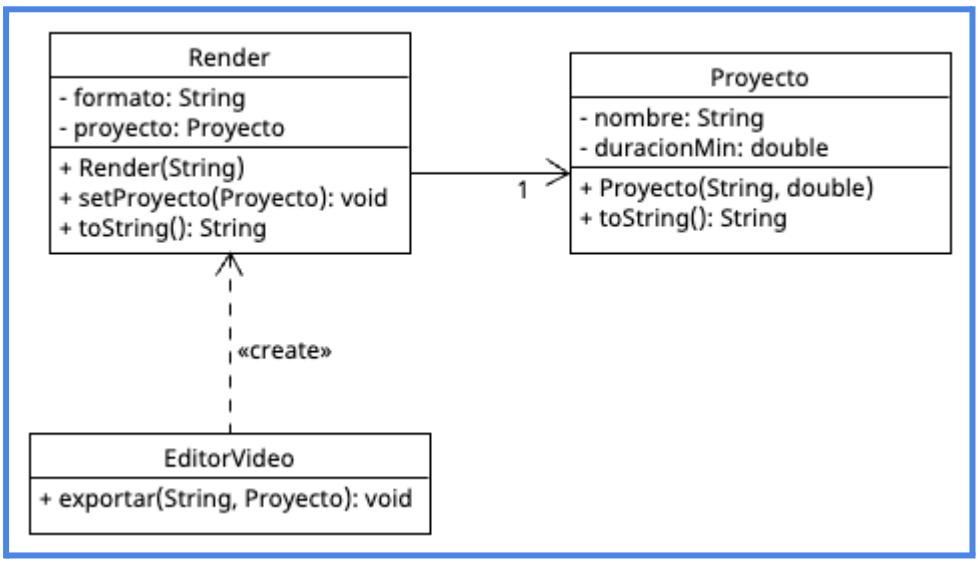
```
run:
QR generado exitosamente para usuario: belenyb
BUILD SUCCESSFUL (total time: 0 seconds)
```

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)

Diagrama UML



Tipo de relación y dirección

- Asociación unidireccional: Render → Proyecto
- Dependencia de creación: EditorVideo.exportar(String, Proyecto)

Implementación de las clases con atributos y relaciones definidas

```java
public class Render {
    private String formato;
    private Proyecto proyecto; // Asociación unidireccional Render -> Proyecto

    public Render(String formato) {
        this.formato = formato;
    }

    public String getFormato() {
        return formato;
    }

    public void setFormato(String formato) {
        this.formato = formato;
    }

    public Proyecto getProyecto() {
        return proyecto;
    }

    public void setProyecto(Proyecto proyecto) {
        this.proyecto = proyecto;
    }

    @Override
    public String toString() {
        return "Render{" + "formato=" + formato + ", proyecto=" + proyecto + '}';
    }

}

public class Proyecto {
    private String nombre;
    private double duracionMin;

    public Proyecto(String nombre, double duracionMin) {
        this.nombre = nombre;
        this.duracionMin = duracionMin;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public double getDuracionMin() {
        return duracionMin;
    }

    public void setDuracionMin(double duracionMin) {
        this.duracionMin = duracionMin;
    }

    @Override
    public String toString() {
        return "Proyecto{" + "nombre=" + nombre + ", duracionMin=" + duracionMin + '}';
    }

}
```

```java
public class EditorVideo {
        // Dependencia de Creación: crea un Render en el método y no lo guarda como atributo.

        public void exportar(String formato, Proyecto proyecto) {
        Render render = new Render(formato);
        render.setProyecto(proyecto); // Asociación unidireccional Render -> Proyecto
        System.out.println("Render generado exitosamente para proyecto: " +
render.getProyecto().getNombre());
    }
}

public class Main {
    public static void main(String[] args) {
        Proyecto proyecto = new Proyecto("Corto A", 50);
        EditorVideo editorVideo = new EditorVideo();
        editorVideo.exportar("mp4", proyecto); // Crea instancia de Render
    }
}
```

Ejecución del código en consola:

```
run:
Render generado exitosamente para proyecto: Corto A
BUILD SUCCESSFUL (total time: 0 seconds)
```