PROGRAMACIÓN II

Trabajo Práctico 5: Relaciones UML 1 a 1

OBJETIVO GENERAL

Modelar clases con relaciones 1 a 1 utilizando diagramas UML. Identificar correctamente el tipo de relación (asociación, agregación, composición, dependencia) y su dirección, y llevarlas a implementación en Java.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Asociación	Relación entre clases con referencia mutua o directa, puede ser uni o bidireccional
Agregación	Relación de "tiene un" donde los objetos pueden vivir independientemente
Composición	Relación fuerte de contención, el ciclo de vida del objeto contenido depende del otro
Dependencia de uso	Una clase usa otra como parámetro en un método, sin almacenarla como atributo
Dependencia de creación	Una clase crea otra en tiempo de ejecución, sin mantenerla como atributo
Asociación	Relación entre clases con referencia mutua o directa, puede ser uni o bidireccional
Agregación	Relación de "tiene un" donde los objetos pueden vivir independientemente

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
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
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

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
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
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
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
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
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

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.

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)
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)

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)

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)

CONCLUSIONES ESPERADAS

- Diferenciar claramente los tipos de relaciones entre clases (asociación, agregación, composición).
- Representar las relaciones con la dirección adecuada en diagramas UML.
- Comprender e implementar dependencias de uso y de creación.
- Aplicar relaciones 1 a 1 en el diseño e implementación de clases en Java.
- Reforzar el análisis de modelos orientados a objetos y la capacidad de abstracción.

RESOLUCIÓN

EJERCICIO 1: Pasaporte - Foto – Titular

1. Clases y Atributos

Pasaporte: numero, fechaEmision.

Foto: imagen, formato.

Titular: nombre, dni.

2. Tipo y Dirección de Relación

Composición: Pasaporte → Foto.

Asociación Bidireccional: Pasaporte ←→ Titular.

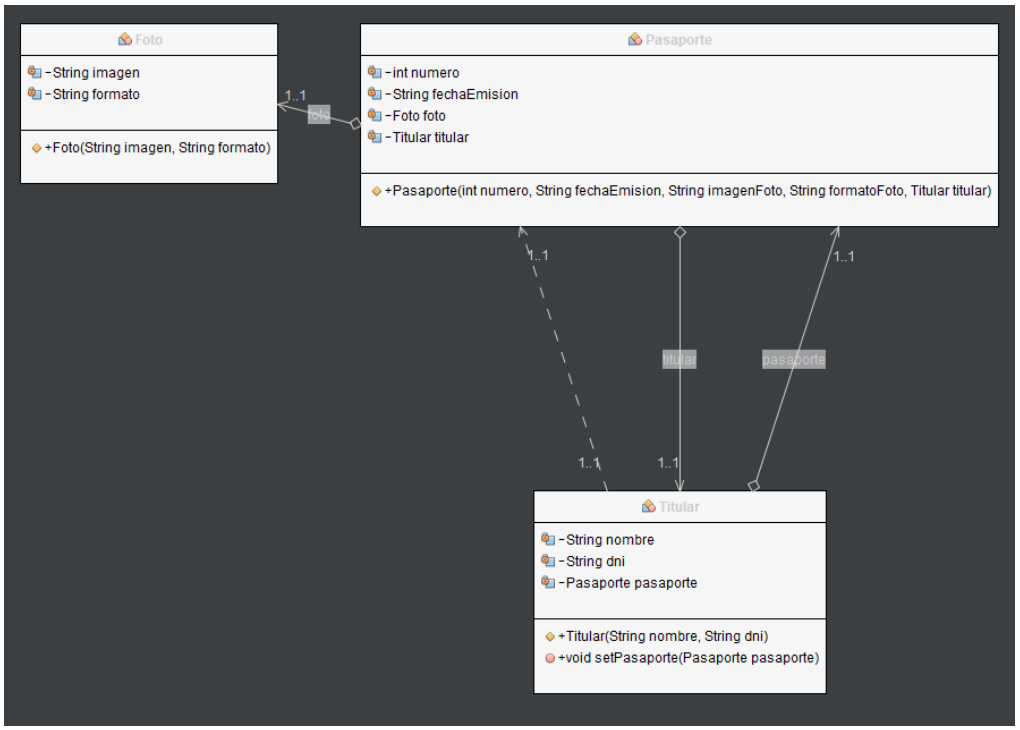
3. Código de cada Clase

```
package Ejercicio1.newpackage;

class Foto {
    private String imagen;
    private String formato;

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

package Ejercicio1.newpackage;

class Titular {
```

```
    private String nombre;
    private String dni;
    private Pasaporte pasaporte;

    public Titular(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setPasaporte(Pasaporte pasaporte) { this.pasaporte = pasaporte;
}
}
```

```
package Ejercicio1.newpackage;

class Pasaporte {
    private int numero;
    private String fechaEmision;
    private Foto foto;
    private Titular titular;

    public Pasaporte(int numero, String fechaEmision, String imagenFoto, String
formatoFoto, Titular titular) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(imagenFoto, formatoFoto);
        this.titular = titular;
        titular.setPasaporte(this);
    }
}
```

4. Diagrama UML



2. EJERCICIO: Celular - Batería – Usuario

1. Clases y Atributos

Celular: imei, marca, modelo.

Batería: modelo, capacidad.

Usuario: nombre, dni.

2. Tipo y Dirección de Relación

Agregación: Celular → Batería.

Asociación Bidireccional: Celular ←→ Usuario.

3. Código de cada Clase

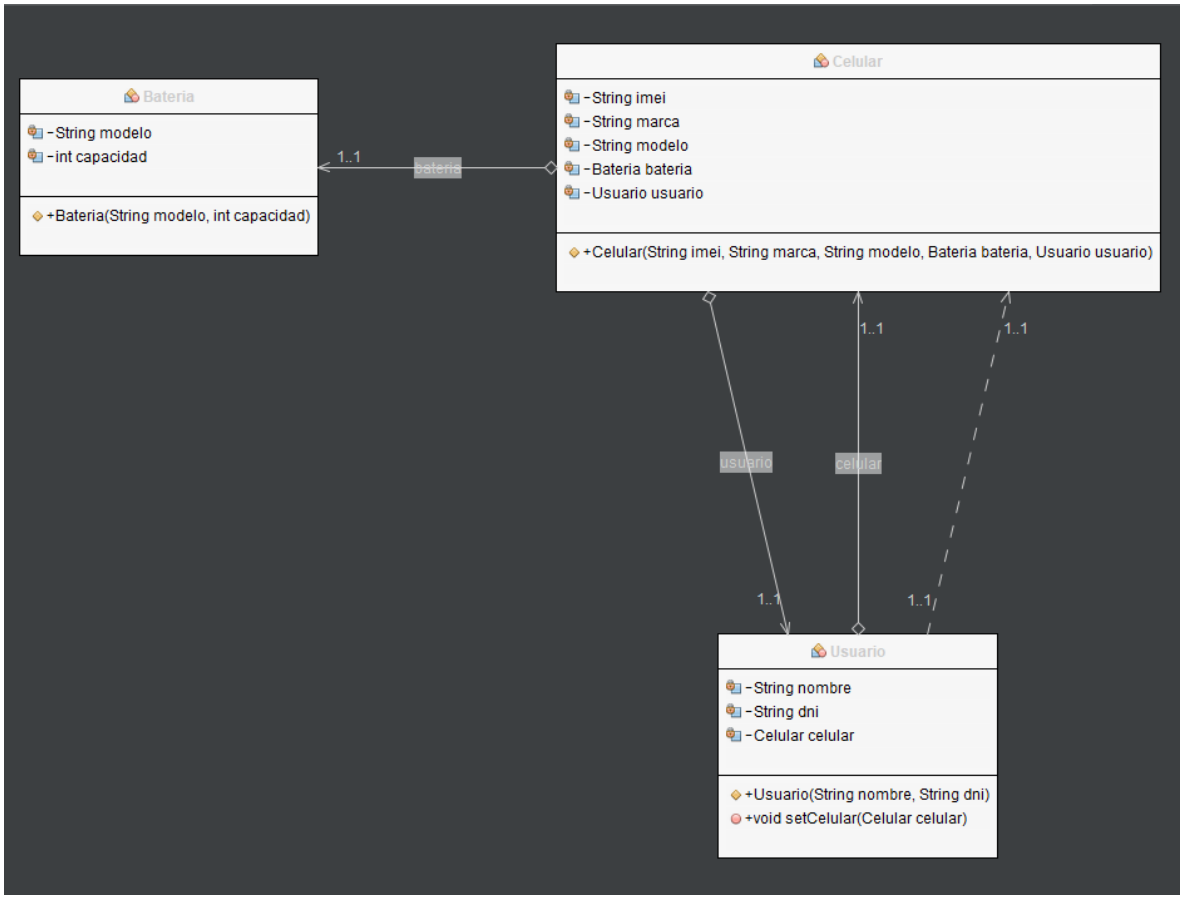
```
package Ejercicio2.newpackage;

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

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

```
package Ejercicio2.newpackage;

public class Usuario {
    private String nombre;
    private String dni;
    // Referencia de vuelta para la Asociación Bidireccional
    private Celular celular;

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

    // Setter para que Celular cierre la relación bidireccional
    public void setCelular(Celular celular) {
        this.celular = celular;
    }
}
```

```
package Ejercicio2.newpackage;

public class Celular {
    private String imei;
    private String marca;
```

```
    private String modelo;

    // Atributo para la Agregación (Batería)
    private Bateria bateria;

    // Atributo para la Asociación Bidireccional (Usuario)
    private Usuario usuario;

    public Celular(String imei, String marca, String modelo, Bateria bateria,
Usuario usuario) {
        this.imei = imei;
        this.marca = marca;
        this.modelo = modelo;

        // IMPLEMENTACIÓN DE LA AGREGACIÓN: Recibe un objeto ya
existente
        this.bateria = bateria;

        // IMPLEMENTACIÓN DE LA ASOCIACIÓN BIDIRECCIONAL: Cierra el
lazo
        this.usuario = usuario;
        usuario.setCelular(this); // Llama al setter en Usuario
    }
}
```

4. Diagrama UML



3. EJERCICIO: Libro - Autor - Editorial

1. Clases y Atributos

Libro: titulo, isbn.

Autor: nombre, nacionalidad.

Editorial: nombre, dirección.

2. Tipo y Dirección de Relación

Asociación Unidireccional: Libro → Autor.

Agregación: Libro → Editorial.

3. Código de cada Clase

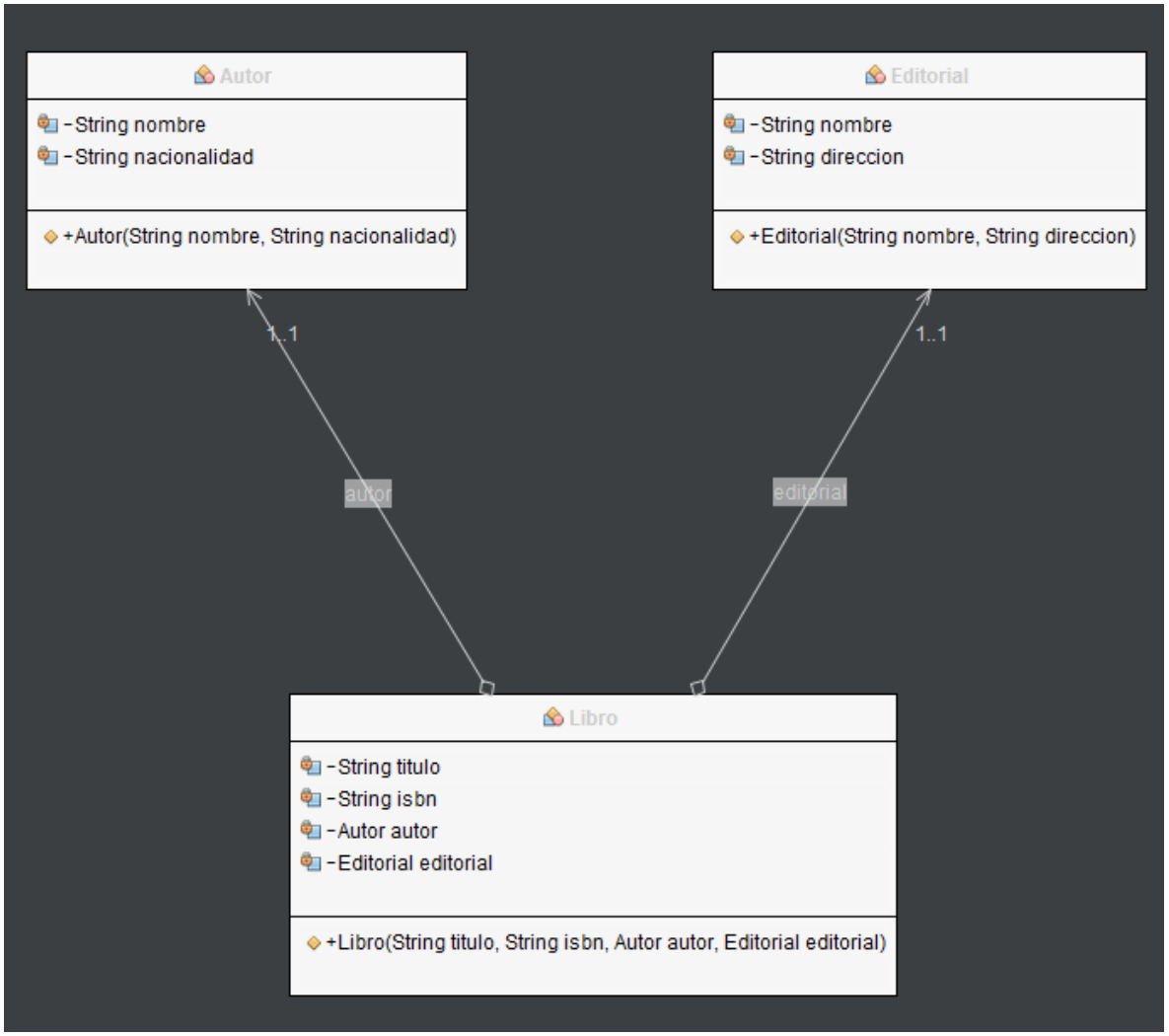
```
package Ejercicio3.newpackage;

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

    public Autor(String nombre, String nacionalidad) {
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }
}
```

```
package Ejercicio3.newpackage;

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

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

```
package Ejercicio3.newpackage;

public class Libro {
    private String titulo;
    private String isbn;
    private Autor autor; // Asociación Unidireccional
    private Editorial editorial; // Agregación

    public Libro(String titulo, String isbn, Autor autor, Editorial editorial) {
        this.titulo = titulo;
        this.isbn = isbn;
        this.autor = autor;
```

```
        this.editorial = editorial;
    }
}
```

4. Diagrama UML



4. EJERCICIO: TarjetaDeCrédito - Cliente - Banco

1. Clases y Atributos

TarjetaDeCrédito: numero, fechaVencimiento.

Cliente: nombre, dni.

Banco: nombre, cuit.

2. Tipo y Dirección de Relación

Asociación Bidireccional: TarjetaDeCrédito ←→ Cliente.

Agregación: TarjetaDeCrédito → Banco.

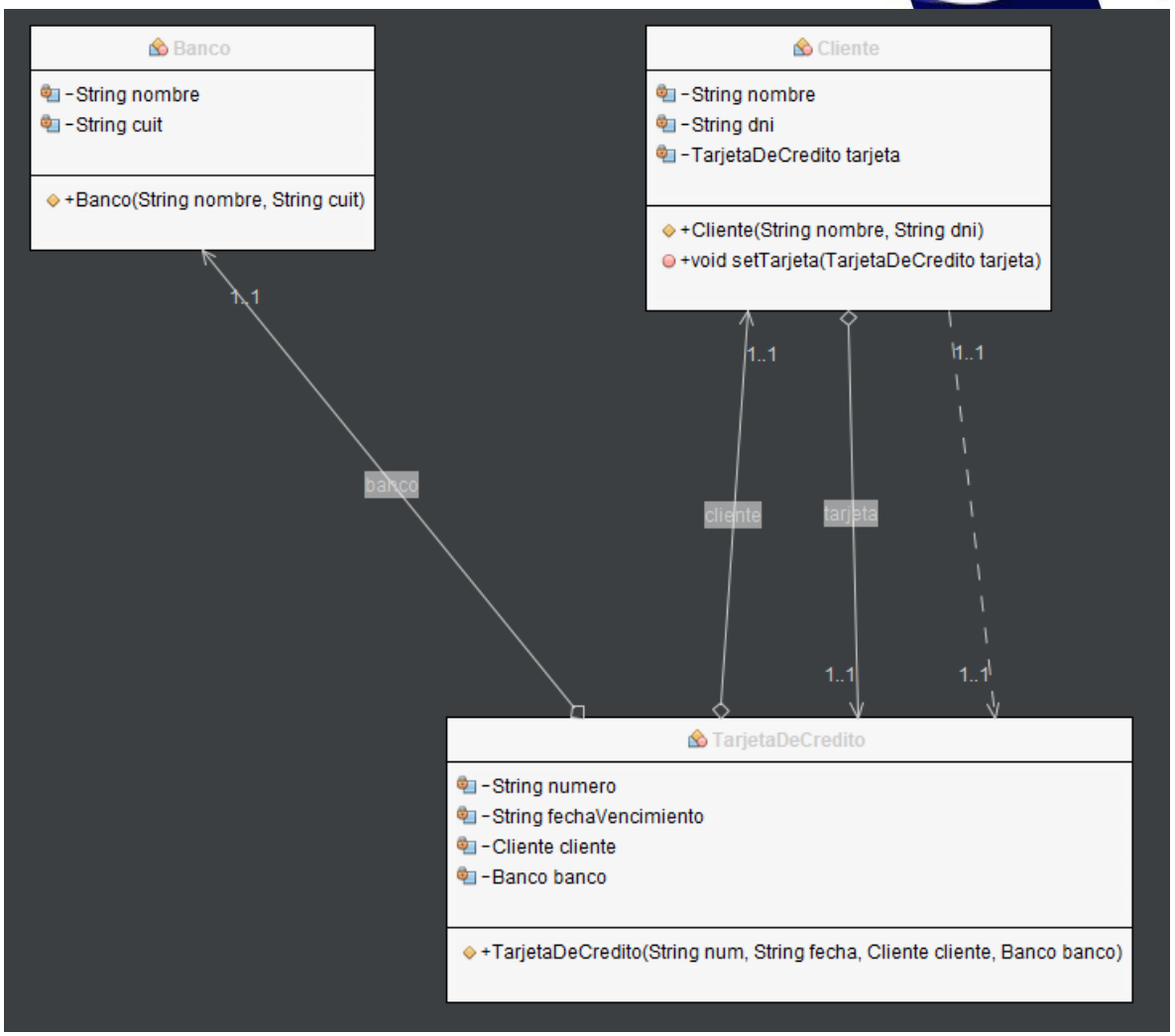
3. Código de cada Clase

```
package Ejercicio4.newpackage;

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

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

```
package Ejercicio4.newpackage;

public class Cliente {
    private String nombre;
    private String dni;
    private TarjetaDeCredito tarjeta;

    public Cliente(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setTarjeta(TarjetaDeCredito tarjeta) {
        this.tarjeta = tarjeta;
    }
}
```

```
package Ejercicio4.newpackage;

public class TarjetaDeCredito {
    private String numero;
    private String fechaVencimiento;
    private Cliente cliente;
    private Banco banco;

    public TarjetaDeCredito(String num, String fecha, Cliente cliente, Banco
banco) {
        this.numero = num;
        this.fechaVencimiento = fecha;
        // Agregación
        this.banco = banco;
        // Bidireccional: cierra el lazo
        this.cliente = cliente;
        cliente.setTarjeta(this);
    }
}
```

4. Diagrama UML



5. EJERCICIO: Computadora - Placa Madre - Propietario

1. Clases y Atributos

Computadora: marca, numeroSerie.

PlacaMadre: modelo, chipset.

Propietario: nombre, dni.

2. Tipo y Dirección de Relación

Composición: Computadora → PlacaMadre.

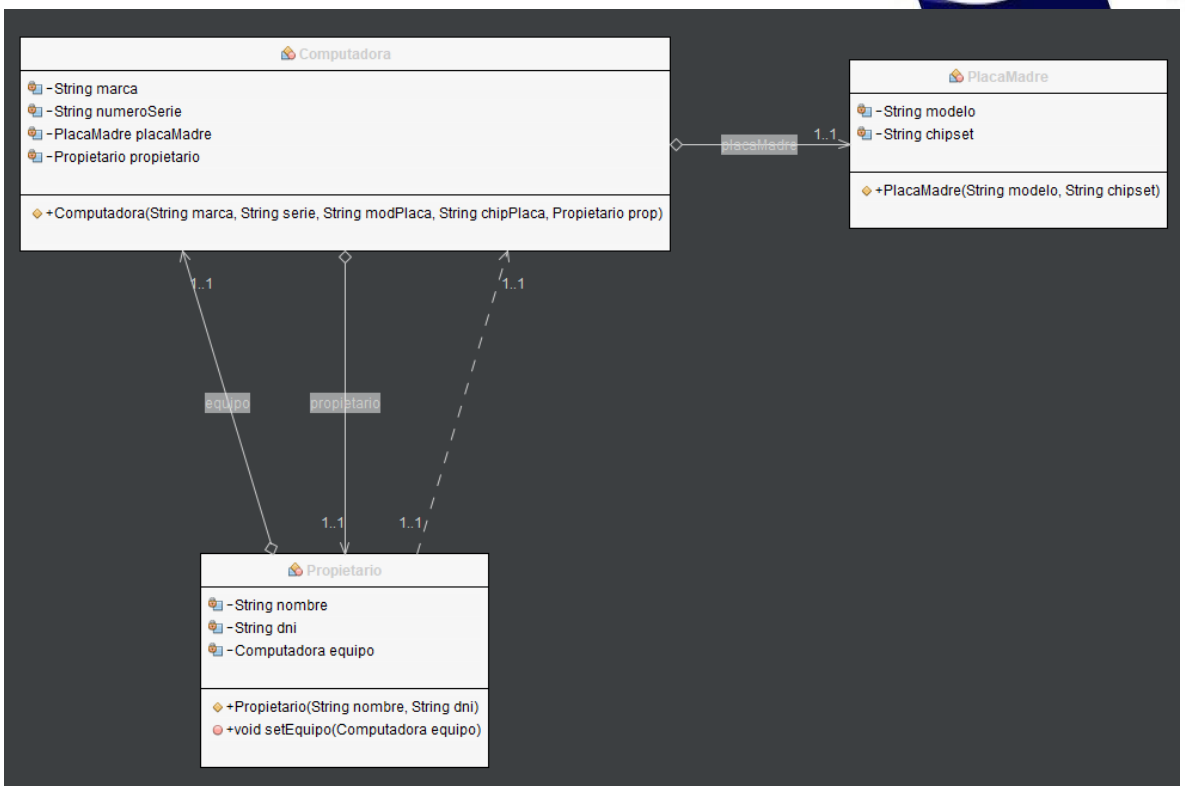
Asociación Bidireccional: Computadora ←→ Propietario.

3. Código de cada Clase

```
package Ejercicio5.newpackage;

public class PlacaMadre {
    private String modelo;
    private String chipset;
```

```
    public PlacaMadre(String modelo, String chipset) {
        this.modelo = modelo;
        this.chipset = chipset;
    }
}

package Ejercicio5.newpackage;

public class Propietario {
    private String nombre;
    private String dni;
    private Computadora equipo;

    public Propietario(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setEquipo(Computadora equipo) {
        this.equipo = equipo;
    }
}

package Ejercicio5.newpackage;

public class Computadora {
    private String marca;
    private String numeroSerie;
    private PlacaMadre placaMadre; // Composición
    private Propietario propietario; // Asociación Bidireccional

    public Computadora(String marca, String serie, String modPlaca, String
chipPlaca, Propietario prop) {
        this.marca = marca;
        this.numeroSerie = serie;
        // Composición: Crea el objeto interno
        this.placaMadre = new PlacaMadre(modPlaca, chipPlaca);
        // Bidireccional: cierra el lazo
        this.propietario = prop;
        prop.setEquipo(this);
    }
}
```

4. Diagrama UML



6. EJERCICIO: Reserva - Cliente - Mesa

1. Clases y Atributos

Reserva: fecha, hora.

Cliente: nombre, telefono.

Mesa: numero, capacidad.

2. Tipo y Dirección de Relación

Asociación Unidireccional: Reserva → Cliente.

Agregación: Reserva → Mesa.

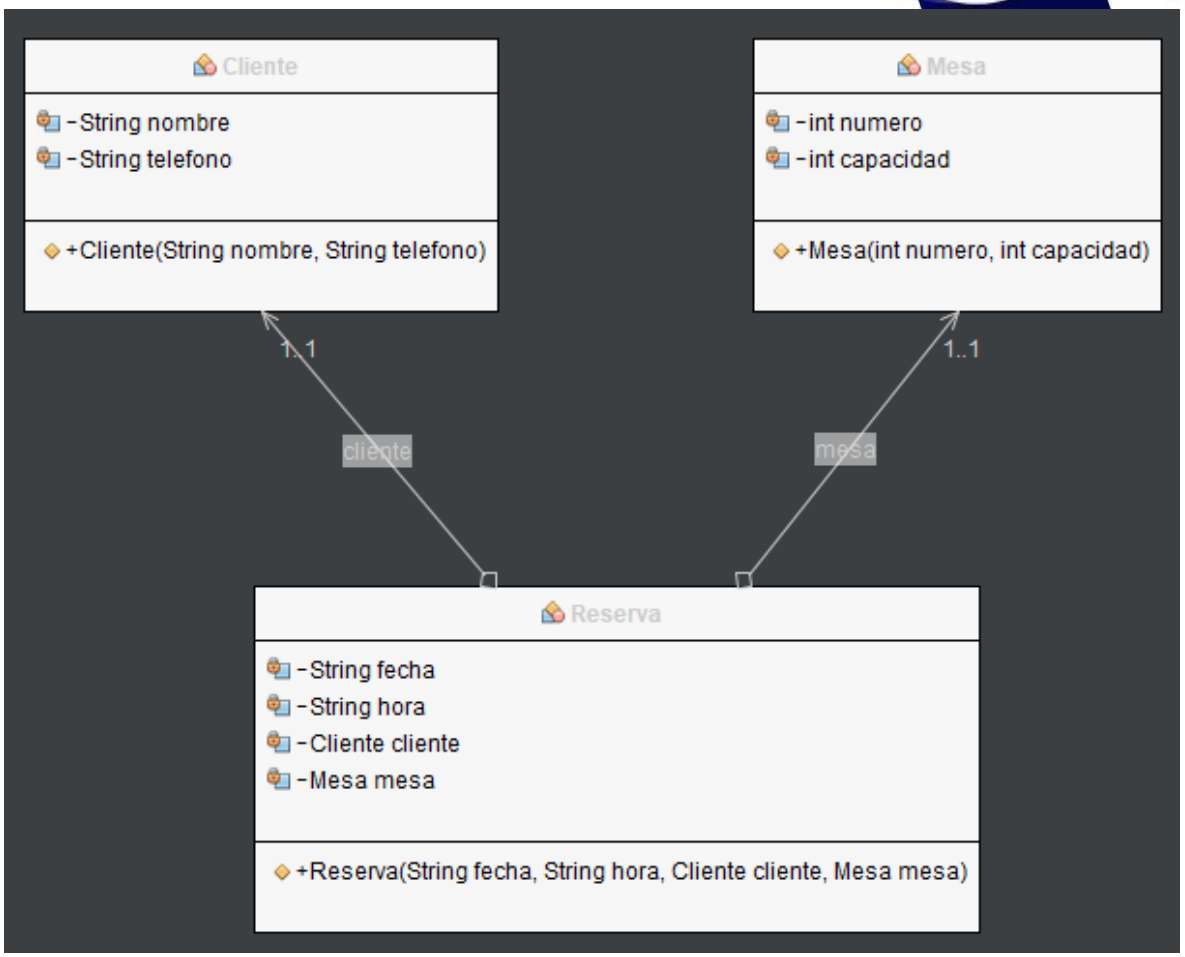
3. Código de cada Clase

```
package Ejercicio6.newpackage;

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

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

```
package Ejercicio6.newpackage;

public class Cliente {
    private String nombre;
    private String telefono;

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

```
package Ejercicio6.newpackage;

public class Reserva {
    private String fecha;
    private String hora;
    private Cliente cliente; // Asociación Unidireccional
    private Mesa mesa; // Agregación

    public Reserva(String fecha, String hora, Cliente cliente, Mesa mesa) {
        this.fecha = fecha;
        this.hora = hora;
        this.cliente = cliente;
        this.mesa = mesa;
    }
}
```

4. Diagrama UML



7. EJERCICIO: Vehículo - Motor - Conductor

1. Clases y Atributos

Vehículo: patente, modelo.

Motor: tipo, numeroSerie.

Conductor: nombre, licencia.

2. Tipo y Dirección de Relación

Agregación: Vehículo → Motor.

Asociación Bidireccional: Vehículo ←→ Conductor.

3. Código de cada Clase

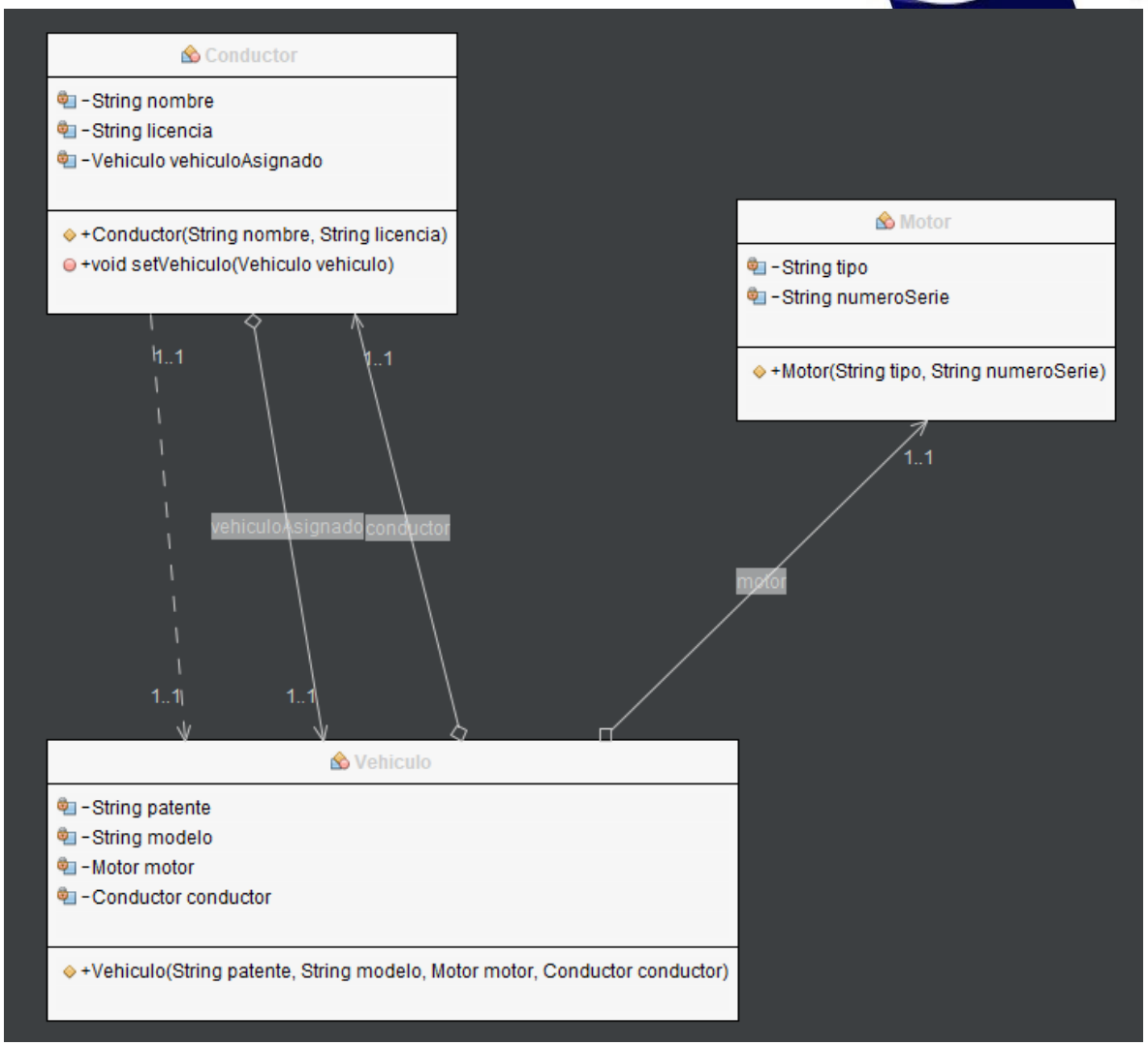
```
package Ejercicio7.newpackage;

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

    public Motor(String tipo, String numeroSerie) {
```

```
        this.tipo = tipo;
        this.numeroSerie = numeroSerie;
    }
}
```

```
package Ejercicio7.newpackage;

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

    public Conductor(String nombre, String licencia) {
        this.nombre = nombre;
        this.licencia = licencia;
    }
    public void setVehiculo(Vehiculo vehiculo) {
        this.vehiculoAsignado = vehiculo;
    }
}
```

```
package Ejercicio7.newpackage;

public class Vehiculo {
    private String patente;
    private String modelo;
    private Motor motor;
    private Conductor conductor;

    public Vehiculo(String patente, String modelo, Motor motor, Conductor
conductor) {
        this.patente = patente;
        this.modelo = modelo;
        // Agregación
        this.motor = motor;
        // Bidireccional: cierra el lazo
        this.conductor = conductor;
        conductor.setVehiculo(this);
    }
}
```

4. Diagrama UML



8. EJERCICIO: Documento - FirmaDigital - Usuario

1. Clases y Atributos

Documento: titulo, contenido.

FirmaDigital: codigoHash, fecha.

Usuario: nombre, email.

2. Tipo y Dirección de Relación

Composición: Documento → FirmaDigital.

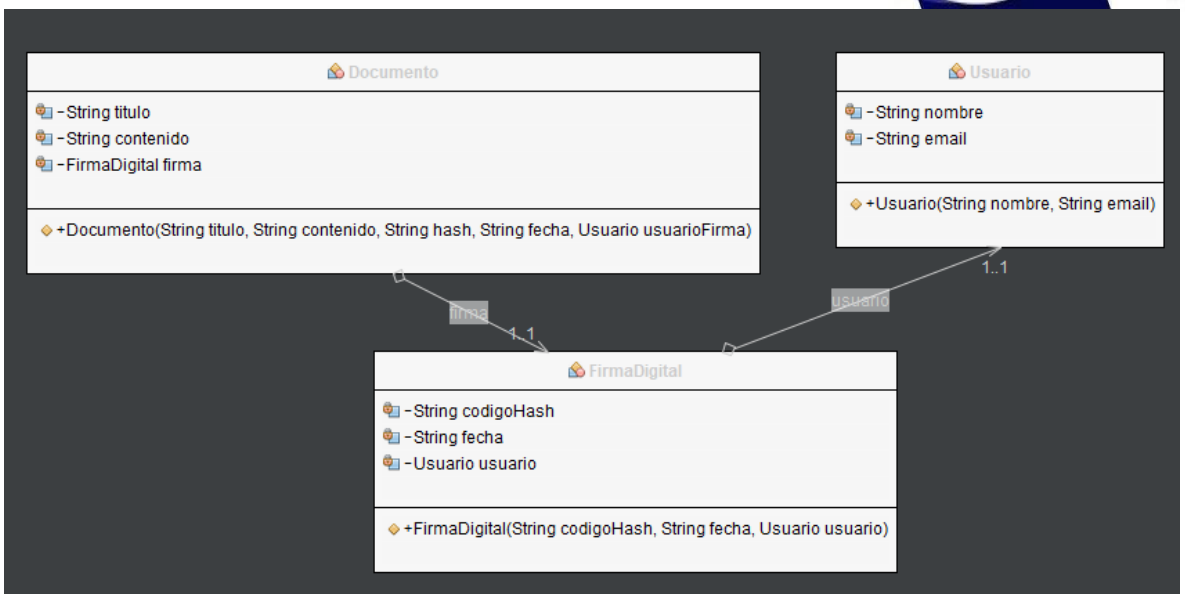
Agregación: FirmaDigital → Usuario.

3. Código de cada Clase

```
package Ejercicio8.newpackage;

public class Usuario {
    private String nombre;
```

```
    private String email;

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

package Ejercicio8.newpackage;

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

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

package Ejercicio8.newpackage;

public class Documento {
    private String titulo;
    private String contenido;
    private FirmaDigital firma; // Composición

    public Documento(String titulo, String contenido, String hash, String fecha,
Usuario usuarioFirma) {
        this.titulo = titulo;
        this.contenido = contenido;
        // Composición: Crea el objeto FirmaDigital
        this.firma = new FirmaDigital(hash, fecha, usuarioFirma);
    }
}
```

4. Diagrama UML



9. EJERCICIO: Cita Médica - Paciente - Profesional

1. Clases y Atributos

CitaMédica: fecha, hora.

Paciente: nombre, obraSocial.

Profesional: nombre, especialidad.

2. Tipo y Dirección de Relación

Asociación Unidireccional: CitaMédica → Paciente.

Asociación Unidireccional: CitaMédica → Profesional.

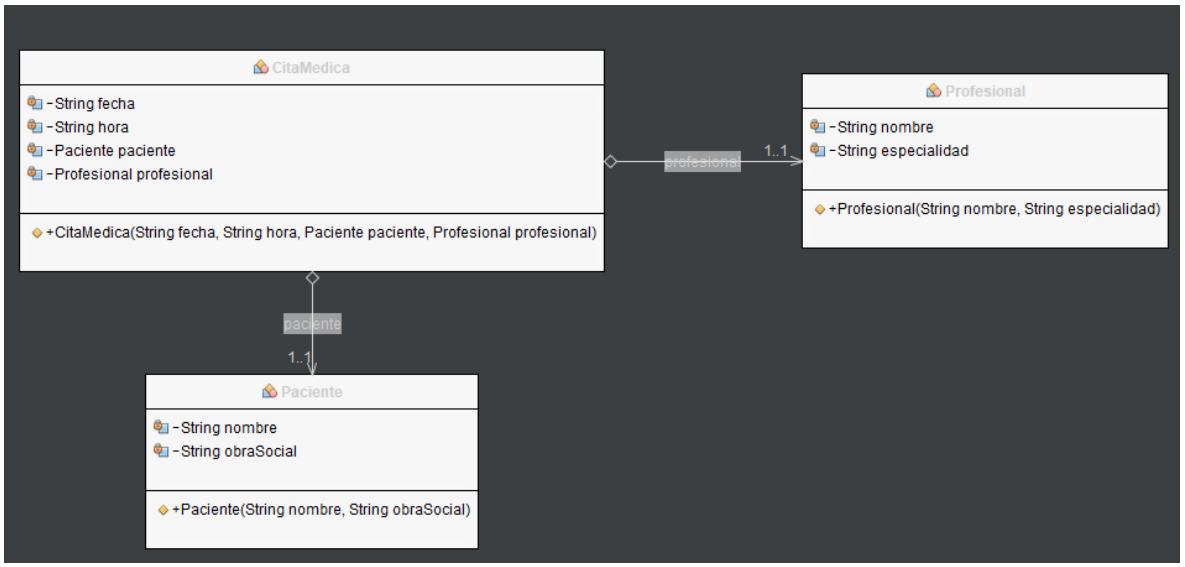
3. Código de cada Clase

```
package Ejercicio9.newpackage;

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

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

```
package Ejercicio9.newpackage;

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

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

package Ejercicio9.newpackage;

public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente; // Asociación Unidireccional
    private Profesional profesional; // Asociación Unidireccional

    public CitaMedica(String fecha, String hora, Paciente paciente, Profesional
profesional) {
        this.fecha = fecha;
        this.hora = hora;
        this.paciente = paciente;
        this.profesional = profesional;
    }
}
```

4. Diagrama UML



10. EJERCICIO: CuentaBancaria - ClaveSeguridad - Titular

1. Clases y Atributos

CuentaBancaria: cbu, saldo.

ClaveSeguridad: codigo, ultimaModificacion.

Titular: nombre, dni.

2. Tipo y Dirección de Relación

Composición: CuentaBancaria → ClaveSeguridad.

Asociación Bidireccional: CuentaBancaria ←→ Titular.

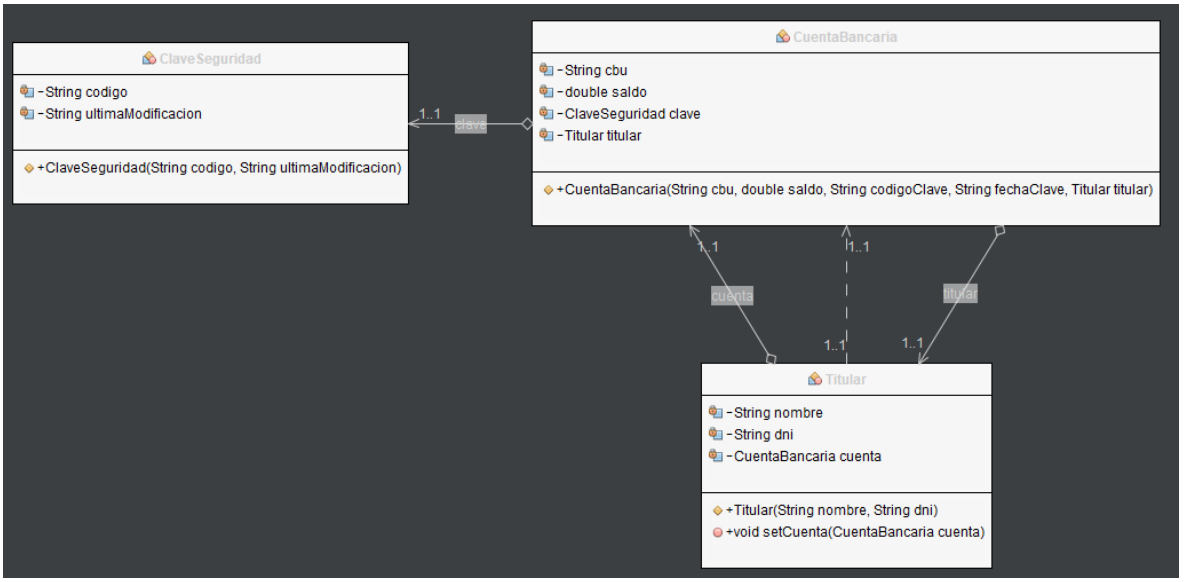
3. Código de cada Clase

```
package Ejercicio10.newpackage;

public class ClaveSeguridad {
    private String codigo;
    private String ultimaModificacion;

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

```
package Ejercicio10.newpackage;

public class Titular {
    private String nombre;
    private String dni;
    private CuentaBancaria cuenta;

    public Titular(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setCuenta(CuentaBancaria cuenta) {
        this.cuenta = cuenta;
    }
}
```

```
package Ejercicio10.newpackage;

public class CuentaBancaria {
    private String cbu;
    private double saldo;
    private ClaveSeguridad clave; // Composición
    private Titular titular; // Asociación Bidireccional

    public CuentaBancaria(String cbu, double saldo, String codigoClave, String
fechaClave, Titular titular) {
        this.cbu = cbu;
        this.saldo = saldo;
        // Composición
        this.clave = new ClaveSeguridad(codigoClave, fechaClave);
        // Bidireccional: cierra el lazo
        this.titular = titular;
        titular.setCuenta(this);
    }
}
```

4. Diagrama UML



11. EJERCICIO: Reproductor - Canción - Artista

1. Clases y Atributos

Canción: titulo.

Artista: nombre, genero.

Reproductor: método void reproducir (Cancion cancion).

2. Tipo y Dirección de Relación

Asociación Unidireccional: Canción → Artista.

Dependencia de Uso: Reproductor → Canción.

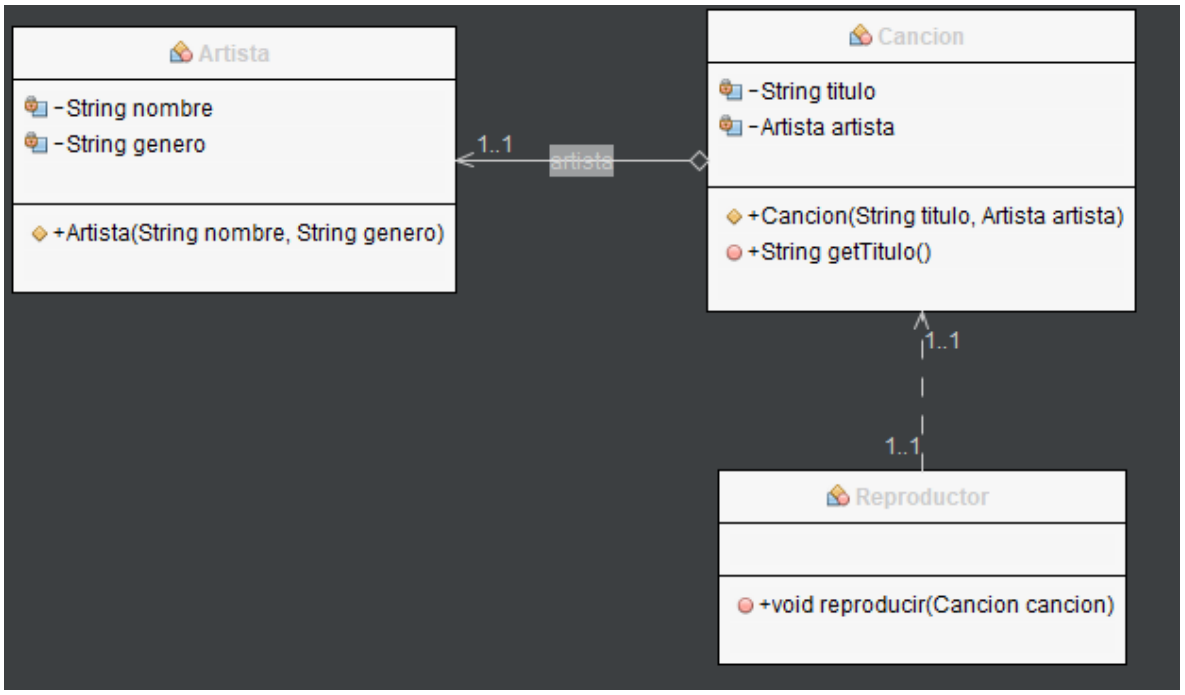
3. Código de cada Clase

```
package Ejercicio11.newpackage;

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

    public Artista(String nombre, String genero) {
        this.nombre = nombre;
        this.genero = genero;
    }
}
```

```
package Ejercicio11.newpackage;

public class Cancion {
```

```
    private String titulo;
    private Artista artista; // Asociación Unidireccional

    public Cancion(String titulo, Artista artista) {
        this.titulo = titulo;
        this.artista = artista;
    }
    // Getter necesario para la Dependencia de Uso
    public String getTitulo() { return titulo; }
}

package Ejercicio11.newpackage;

public class Reproductor {
    // Dependencia de Uso: Usa Cancion como parámetro del método
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciendo: " + cancion.getTitulo());
    }
}
```

4. Diagrama UML



12. EJERCICIO: Impuesto - Contribuyente - Calculadora

1. Clases y Atributos

Impuesto: monto.

Contribuyente: nombre, cuil.

Calculadora: método void calcular(Impuesto impuesto).

2. Tipo y Dirección de Relación

Asociación Unidireccional: Impuesto → Contribuyente.

Dependencia de Uso: Calculadora → Impuesto.

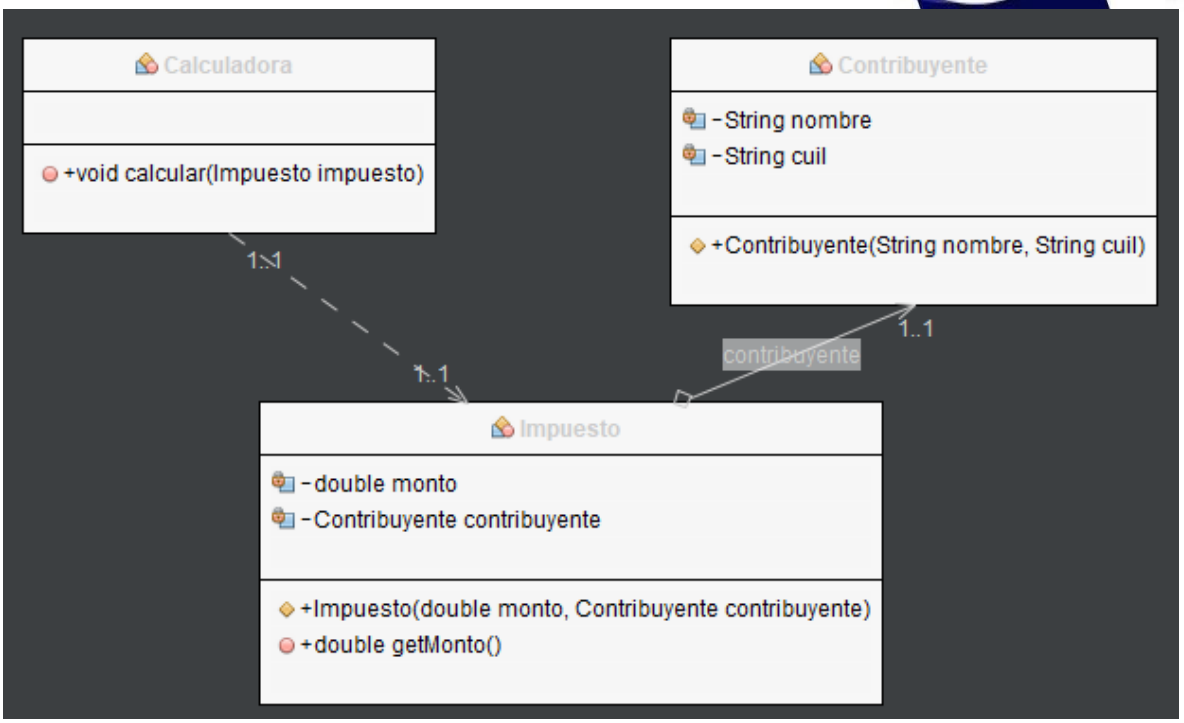
3. Código de cada Clase

```
package Ejercicio12.newpackage;

public class Contribuyente {
    private String nombre;
    private String cuil;

    public Contribuyente(String nombre, String cuil) {
        this.nombre = nombre;
        this.cuil = cuil;
    }
}
```

```
package Ejercicio12.newpackage;

public class Impuesto {
    private double monto;
    private Contribuyente contribuyente; // Asociación Unidireccional

    public Impuesto(double monto, Contribuyente contribuyente) {
        this.monto = monto;
        this.contribuyente = contribuyente;
    }
    // Getter necesario para la Dependencia de Uso
    public double getMonto() { return monto; }
}
```

```
package Ejercicio12.newpackage;

public class Calculadora {
    // Dependencia de Uso: Usa Impuesto como parámetro del método
    public void calcular(Impuesto impuesto) {
        System.out.println("Calculando impuesto de: $" + impuesto.getMonto());
    }
}
```

4. Diagrama UML



13. EJERCICIO: GeneradorQR - Usuario - CódigoQR

1. Clases y Atributos

CódigoQR: valor.

Usuario: nombre, email.

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

2. Tipo y Dirección de Relación

Asociación Unidireccional: CódigoQR → Usuario.

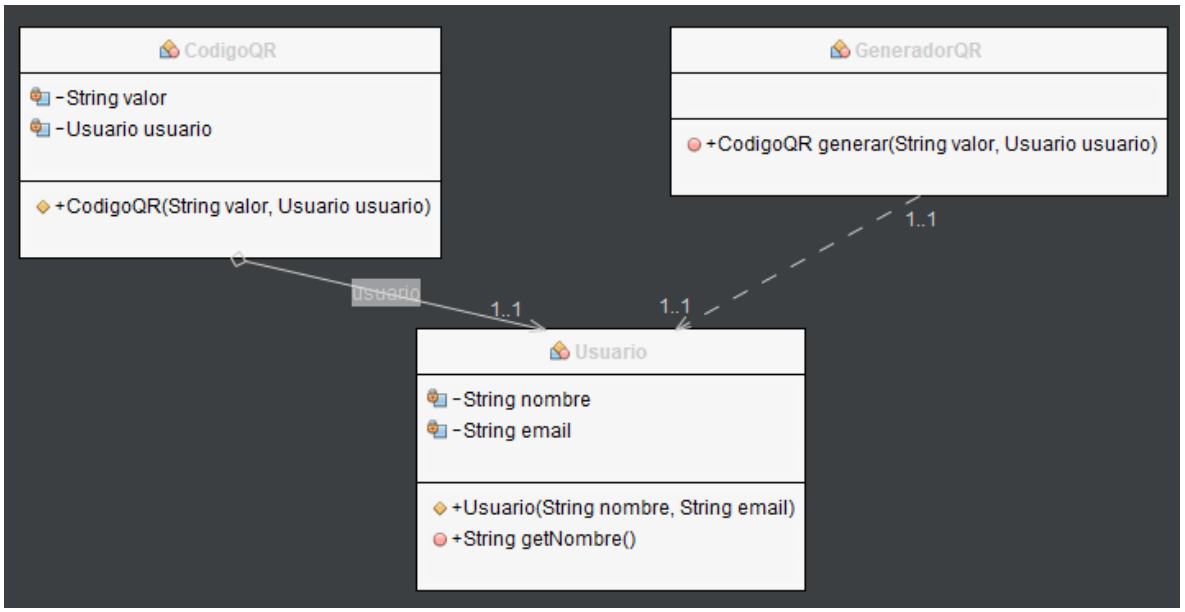
Dependencia de Creación: GeneradorQR → CódigoQR.

3. Código de cada Clase

```
package Ejercicio13.newpackage;

public class CodigoQR {
    private String valor;
    private Usuario usuario; // Asociación Unidireccional

    public CodigoQR(String valor, Usuario usuario) {
        this.valor = valor;
        this.usuario = usuario;
    }
}

package Ejercicio13.newpackage;
```

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

    public Usuario(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }
    // Getter necesario para la Dependencia de Creación
    public String getNombre() { return nombre; }
}

package Ejercicio13.newpackage;

public class GeneradorQR {
    // Dependencia de Creación: Crea CodigoQR dentro del método
    public CodigoQR generar(String valor, Usuario usuario) {
        // Genera la instancia dentro del método
        CodigoQR qr = new CodigoQR(valor, usuario);
        System.out.println("QR generado para: " + usuario.getNombre());
        return qr;
    }
}
```

4. Diagrama UML



14. EJERCICIO: EditorVideo - Proyecto - Render

1. Clases y Atributos

Render: formato.

Proyecto: nombre, duracionMin.

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

2. Tipo y Dirección de Relación

Asociación Unidireccional: Render → Proyecto.

Dependencia de Creación: EditorVideo → Render.

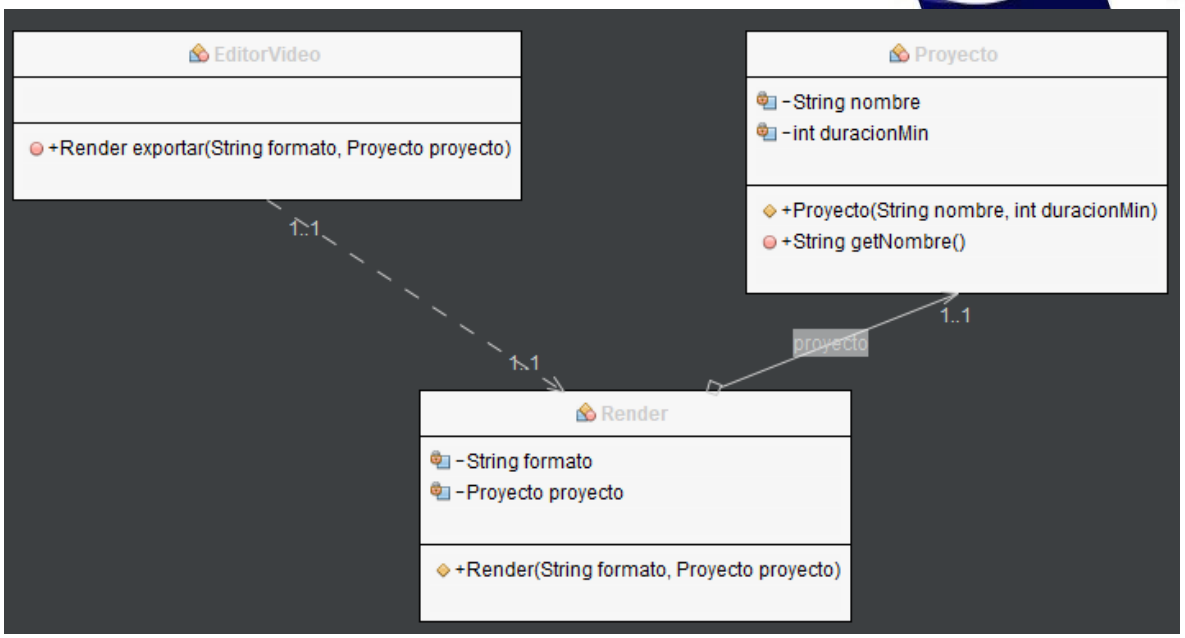
3. Código de cada Clase

```
package Ejercicio14.newpackage;

public class Render {
    private String formato;
    private Proyecto proyecto; // Asociación Unidireccional

    public Render(String formato, Proyecto proyecto) {
        this.formato = formato;
        this.proyecto = proyecto;
    }
}
```

```
package Ejercicio14.newpackage;

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

    public Proyecto(String nombre, int duracionMin) {
        this.nombre = nombre;
        this.duracionMin = duracionMin;
    }
    // Getter necesario para la Dependencia de Creación
    public String getNombre() { return nombre; }
}
```

```
package Ejercicio14.newpackage;

public class EditorVideo {
    // Dependencia de Creación: Crea Render dentro del método
    public Render exportar(String formato, Proyecto proyecto) {
        // Genera la instancia dentro del método
        Render render = new Render(formato, proyecto);
        System.out.println("Exportando proyecto: " + proyecto.getNombre());
        return render;
    }
}
```

4. Diagrama UML



Link de repo: https://github.com/Dario-Cabrera/UTN-TUPaD-P2.git

Alumno: Cabrera Dario Ezequiel

DNI: 41375492