PROGRAMACIÓN II
Trabajo Práctico 5: Relaciones UML 1 a 1

Alumno: Mauro Gaspar
Comisión: 4

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

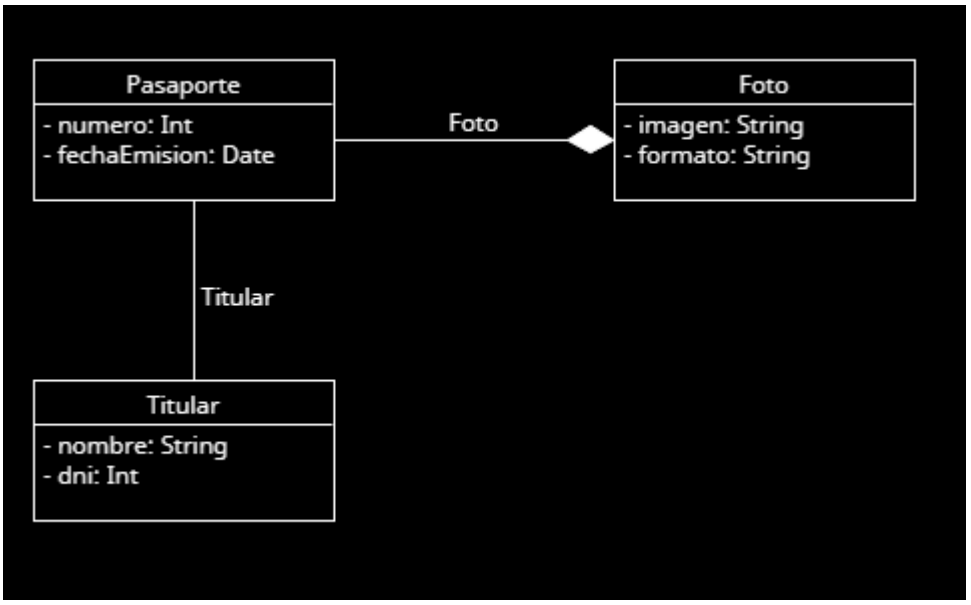
1. Pasaporte - Foto - Titular
    a. Composición: Pasaporte → Foto
    b. Asociación bidireccional: Pasaporte ↔ Titular

Clases y atributos:

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

```
 * @author Mauro
 */
public class MainPasaporte {

    public static void main(String[] args) {
        Titular tl = new Titular ("Mauro Eleazar", "21245453");

        Pasaporte pl = new Pasaporte("AR-2055", LocalDate.of(2025, 2, 2), "mau_foto.png", "PNG");

        pl.setTitular(tl);

        // Mostrar resultados
        System.out.println(pl);
        System.out.println("Titular del Pasaporte: " + pl.getTitular());
        System.out.println("Pasaporte del titular: " + tl.getPasaporte());
    }
}
```

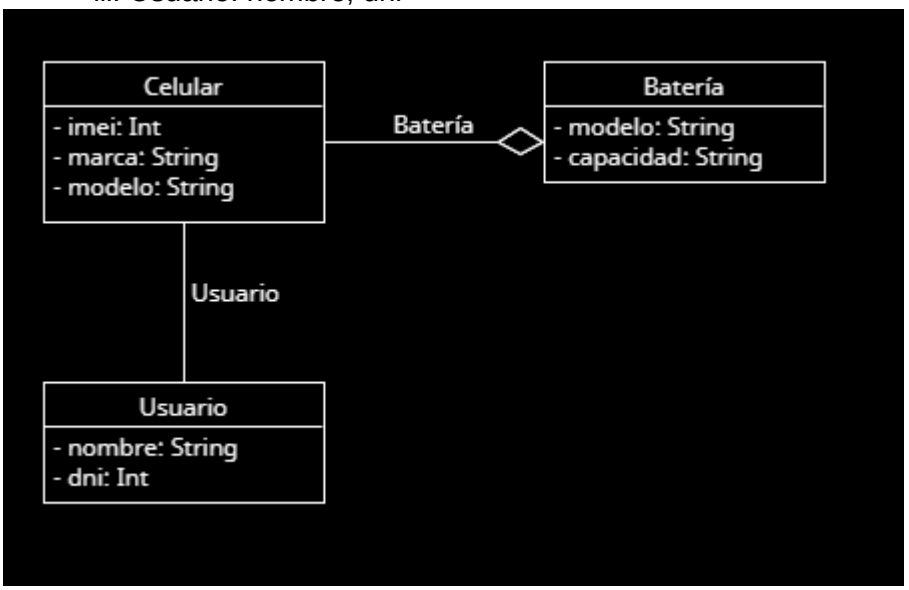
```
run:
Pasaporte [numero=AR-2055, fechaEmision=2025-02-02, Foto [imagen=mau_foto.png, formato=PNG]]
Titular del Pasaporte: Titular [nombre=Mauro Eleazar, dni=21245453]
Pasaporte del titular: Pasaporte [numero=AR-2055, fechaEmision=2025-02-02, Foto [imagen=mau_foto.png, formato=PNG]]
BUILD SUCCESSFUL (total time: 4 seconds)
```



2. Celular - Batería - Usuario
    a. Agregación: Celular → Batería
    b. Asociación bidireccional: Celular ↔ Usuario

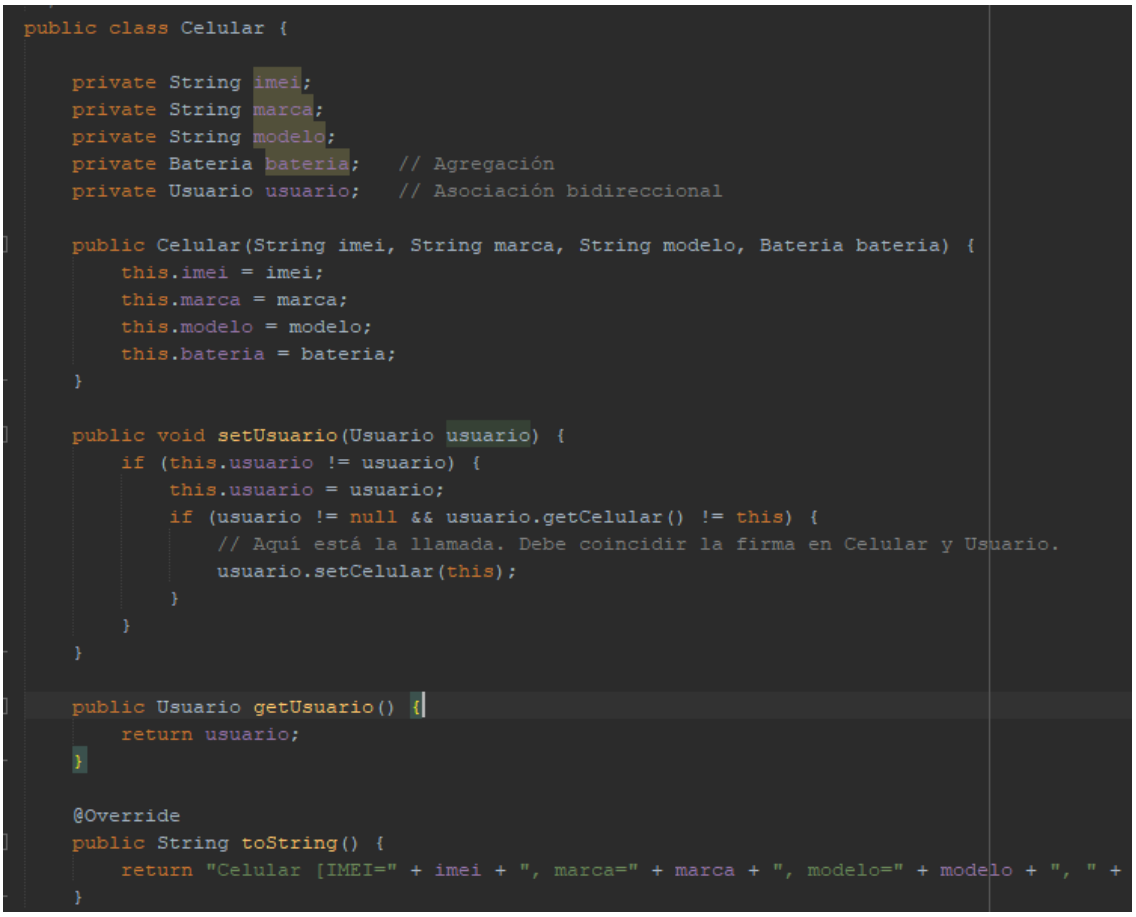
Clases y atributos:

    i. Celular: imei, marca, modelo
    ii. Batería: modelo, capacidad
    iii. Usuario: nombre, dni



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

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

    public String getModelo() {
        return modelo;
    }

    public int getCapacidad() {
        return capacidad;
    }

    @Override
    public String toString() {
        return "Bateria [modelo=" + modelo + ", capacidad=" + capacidad + "mAh]";
    }
}
```

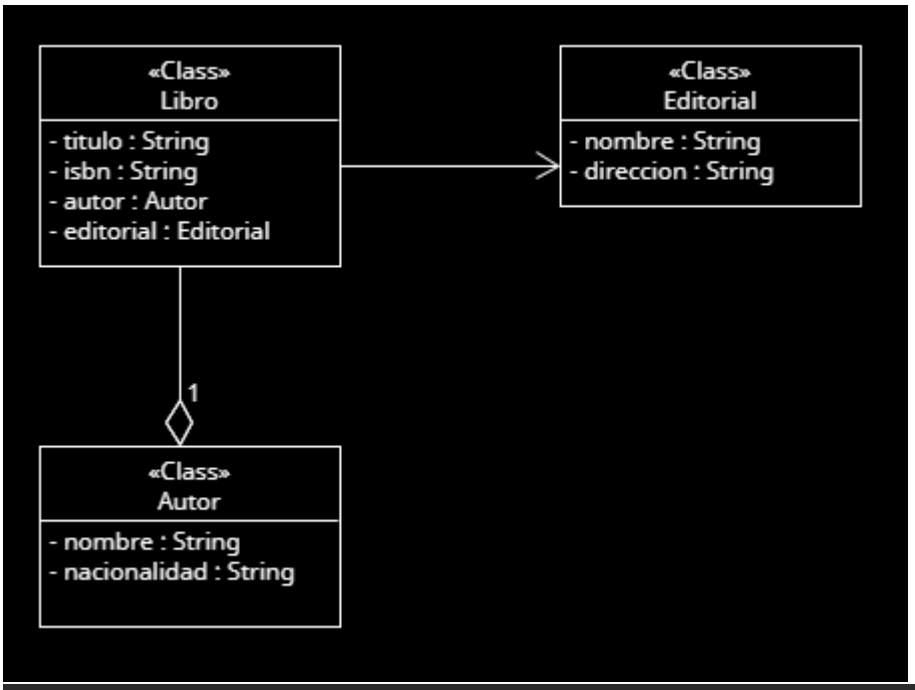
```java
public class Usuario {
    private String nombre;
    private String dni;
    private Celular celular; // Asociaciín bidireccional

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

    public void setCelular(Celular celular) {
        this.celular = celular;
    }

    public Celular getCelular() {
        return celular;
    }

    @Override
    public String toString() {
        return "Usuario [nombre=" + nombre + ", dni=" + dni + "]";
    }
}
```



```java
public class Celular {

    private String imei;
    private String marca;
    private String modelo;
    private Bateria bateria;    // Agregación
    private Usuario usuario;    // Asociación bidireccional

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

    public void setUsuario(Usuario usuario) {
        if (this.usuario != usuario) {
            this.usuario = usuario;
            if (usuario != null && usuario.getCelular() != this) {
                // Aquí está la llamada. Debe coincidir la firma en Celular y Usuario.
                usuario.setCelular(this);
            }
        }
    }

    public Usuario getUsuario() {
        return usuario;
    }

    @Override
    public String toString() {
        return "Celular [IMEI=" + imei + ", marca=" + marca + ", modelo=" + modelo + ", " +
    }
```

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




```java
public class Autor {

    private String nombre;
    private String nacionalidad;

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

    public String getNombre() {
        return nombre;
    }

    public String getNacionalidad() {
        return nacionalidad;
    }

    @Override
    public String toString() {
        return "Autor [nombre=" + nombre + ", nacionalidad=" + nacionalidad + "]";
    }
}
```

```java
public class Libro {

    private String titulo;
    private String isbn;
    private Autor autor;
    private Editorial editorial;

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

    public String getTitulo() {
        return titulo;
    }

    public String getIsbn() {
        return isbn;
    }

    public Autor getAutor() {
        return autor;
    }
```

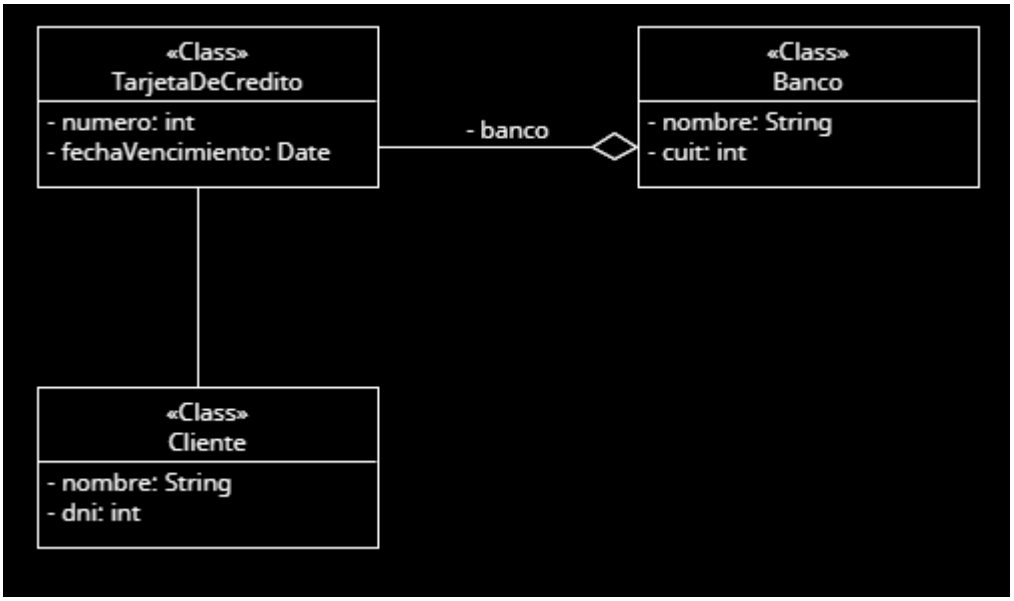
```java
public class Editorial {

    private String nombre;
    private String direccion;

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

    public String getNombre() {
        return nombre;
    }

    public String getDireccion(){
        return direccion;
    }

    @Override
    public String toString() {
        return "Editorial [nombre=" + nombre + ", direccion=" + direccion + "]";
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Autor autor = new Autor("Axel Kaiser", "Chileno");
        Editorial editorial = new Editorial("Ariel", "2024");

        Libro libro = new Libro("Parásitos Mentelas", "123-154-484-55", autor, editorial);

        System.out.println(libro);
    }
}
```

```
Debugger Console × TP5 (run) ×
run:
Libro [titulo= Parásitos Mentelas, isbn = 123-154-484-55
Autor [nombre=Axel Kaiser, nacionalidad=Chileno], Editorial [nombre=Ariel, direccion=2024]]
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



```java
public class Cliente {

    private String nombre, dni;
    private TarjetaDeCredito tarjeta; // Asociación bidireccional

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

    public void setTarjeta(TarjetaDeCredito tarjeta) {
        this.tarjeta = tarjeta;
    }

    public TarjetaDeCredito getTarjeta() {
        return tarjeta;
    }
    @Override
    public String toString() {
        return "Cliente [nombre=" + nombre + ", dni=" + dni + "]";
    }
}
```

```java
public class Banco {

    private String nombre, cuit;

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

    public String getNombre() {
        return nombre;
    }

    public String getCuit () {
        return cuit;
    }

    @Override
    public String toString() {
        return "Banco [nombre=" + nombre + ", cuit=" + cuit + "]";
    }
}
```

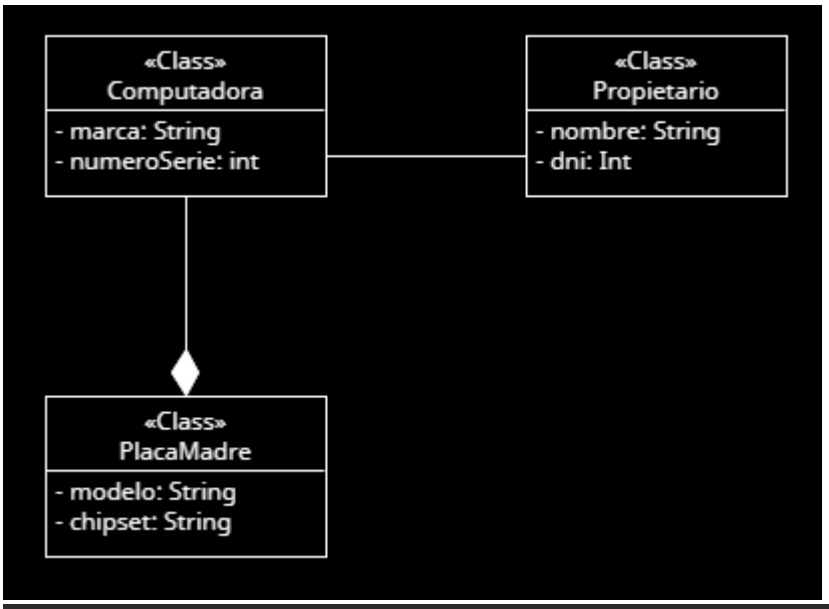
```java
public class TarjetaDeCredito {

    private String numero;
    private LocalDate fechaVencimiento;
    private Cliente cliente;    // Asociación bidireccional
    private Banco banco;    // Agregación

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

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
        cliente.setTarjeta(this); // Mantiene la bidereccionalidad
    }

    public Cliente getCliente() {
        return cliente;
    }

    public Banco getBanco() {
        return banco;
    }

    @Override
    public String toString() {
        return "TarjetaDeCredito [numero=" + numero + ", fechaVencimiento=" + fechaVencimiento + ", " +
    }
```

```java
public class Main {

    public static void main(String[] args) {
        Banco banco = new Banco("Banco Cripto", "66-21279266-6");
        Cliente cliente = new Cliente("Eleazar Gass", "41610272");
        TarjetaDeCredito tarjeta = new TarjetaDeCredito("6485-2127-9298-0513", LocalDate.of(2033, 6, 06),

        // Vincular relaciones
        tarjeta.setCliente(cliente);

        // Mostrar resultados
        System.out.println(tarjeta);
        System.out.println("Cliente de la tarjeta: " + tarjeta.getCliente());
        System.out.println("Banco de la tarjeta: " + tarjeta.getBanco());
        System.out.println("Tarjeta del cliente: " + cliente.getTarjeta());

    }
}
```

```
out ×
Debugger Console ×   TP5 (run) ×
run:
TarjetaDeCredito [numero=6485-2127-9298-0513, fechaVencimiento=2033-06-06, Banco [nombre=Banco Cripto, cuit=66-21279266-6]]
Cliente de la tarjeta: Cliente [nombre=Eleazar Gass, dni=41610272]
Banco de la tarjeta: Banco [nombre=Banco Cripto, cuit=66-21279266-6]
Tarjeta del cliente: TarjetaDeCredito [numero=6485-2127-9298-0513, fechaVencimiento=2033-06-06, Banco [nombre=Banco Cripto,
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




```java
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;
    }

    @Override
    public String toString() {
        return "PlacaMadre [modelo=" + modelo + ", chipset=" + chipset + "]";
    }
}
```

```java
public class Propietario {

    private String nombre;
    private int dni;
    private Computadora computadora; // Asociación bidireccional

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

    public void setComputadora(Computadora computadora) {
        this.computadora = computadora;
    }

    public Computadora getComputadora() {
        return computadora;
    }

    @Override
    public String toString() {
        return "Propietario [nombre=" + nombre + ", dni=" + dni + "]";
    }
}
```

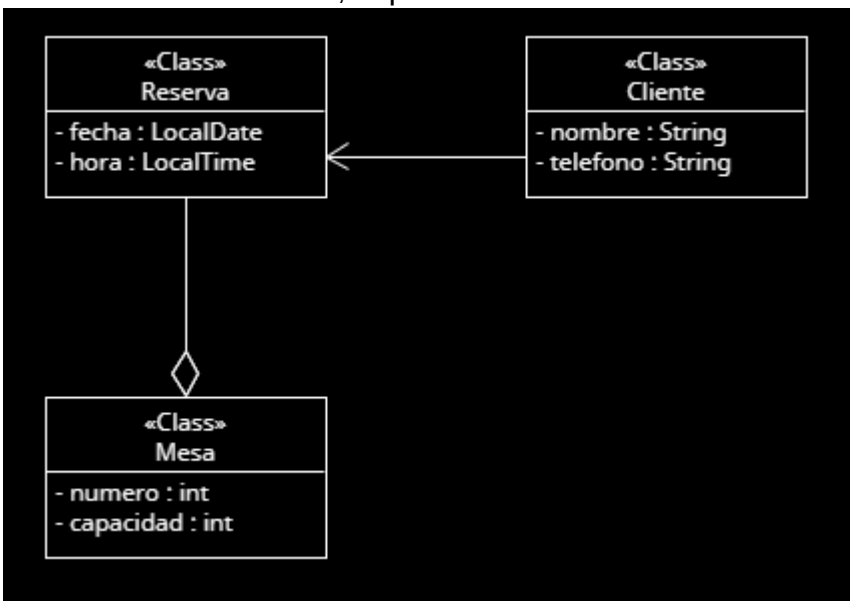
```java
public class Computadora {
    private String marca;
    private String numeroSerie;
    private PlacaMadre placaMadre;   // Composición
    private Propietario propietario; // Asociación bidireccional

    public Computadora(String marca, String numeroSerie, String modeloPlaca, String chipset) {
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        // Composición: la placa madre se crea dentro de la computadora
        this.placaMadre = new PlacaMadre(modeloPlaca, chipset);
    }

    public void setPropietario(Propietario propietario) {
        this.propietario = propietario;
        propietario.setComputadora(this); // Mantiene la bidireccionalidad
    }

    public Propietario getPropietario() {
        return propietario;
    }

    public PlacaMadre getPlacaMadre() {
        return placaMadre;
    }

    @Override
    public String toString() {
        return "Computadora [marca=" + marca + ", numeroSerie=" + numeroSerie + ", " + placaMadr
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        // Crear propietario
        Propietario propietario = new Propietario("Daren Guerrero", 39116609);

        // Crear computadora (compone su placa madre)
        Computadora compu = new Computadora("ASUS", "F16", "Core 5 210H", "H-610");

        // Asociar propietario ↔ computadora
        compu.setPropietario(propietario);

        // Mostrar resultados
        System.out.println(compu);
        System.out.println("Propietario de la computadora: " + compu.getPropietario());
        System.out.println("Computadora del propietario: " + propietario.getComputadora());
    }
}
```

```
ejercicio5.Main > main >
ut ×
Debugger Console ×   TP5 (run) ×
run:
Computadora [marca=ASUS, numeroSerie=F16, PlacaMadre [modelo=Core 5 210H, chipset=H-610]]
Propietario de la computadora: Propietario [nombre=Daren Guerrero, dni=39116609]
Computadora del propietario: Computadora [marca=ASUS, numeroSerie=F16, PlacaMadre [modelo=Core 5 210H, chipset=H-610]]
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



```java
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;
    }

    @Override
    public String toString() {
        return "Cliente [nombre= " + nombre + ", teléfono= " + telefono + "]";
    }
}
```

```java
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;
    }

    @Override
    public String toString() {
        return "Mesa [numero=" + numero + ", capacidad=" + capacidad + "]";
    }
}
```

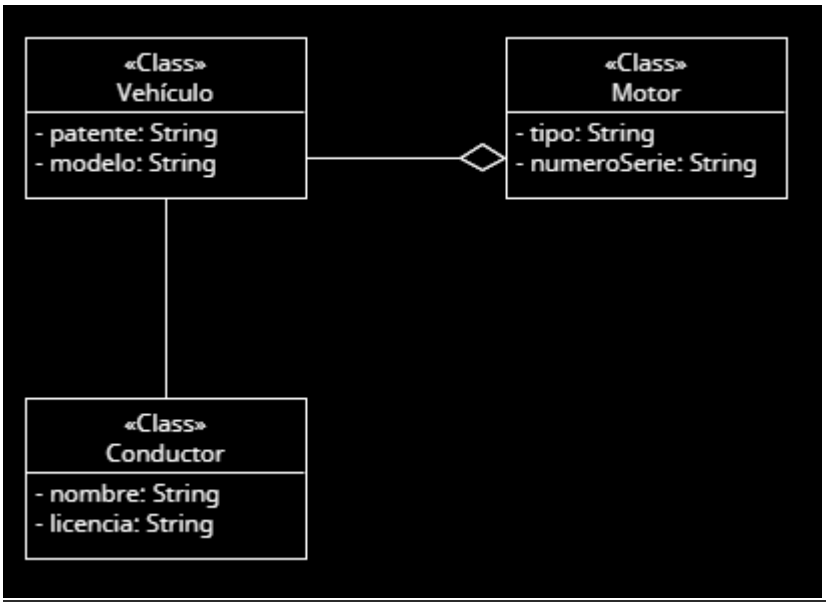
```java
public class Reserva {

    private LocalDate fecha;
    private LocalTime hora;
    private Cliente cliente; // Asociación unidireccional
    private Mesa mesa;       // Agregación

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

    public LocalDate getFecha() {
        return fecha;
    }

    public LocalTime getHora() {
        return hora;
    }

    public Cliente getCliente() {
        return cliente;
    }

    public Mesa getMesa() {
        return mesa;
    }

    @Override
    public String toString() {
        return "Reserva [fecha= " + fecha + ", hora= " + hora + ", " + cliente + ", " + mesa
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        Cliente cliente = new Cliente("Clarita Luz", "98051321");
        Mesa mesa = new Mesa(19, 4);
        Reserva reserva = new Reserva(LocalDate.of(2025, 11, 21), LocalTime.of(21, 0), cliente, mesa);

        System.out.println(reserva);
        System.out.println("Cliente de la reserva: " + reserva.getCliente());
        System.out.println("Mesa reservada: " + reserva.getMesa());
    }
}
```

```
ut
Debugger Console ×   TP5 (run) ×
run:
Reserva [fecha= 2025-11-21, hora= 21:00, Cliente [nombre= Clarita Luz, teléfono= 98051321], Mesa [numero=19, capacidad=4]]
Cliente de la reserva: Cliente [nombre= Clarita Luz, teléfono= 98051321]
Mesa reservada: Mesa [numero=19, capacidad=4]
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




```java
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;
    }

    @Override
    public String toString() {
        return "Motor [tipo = " + tipo + ", numeroSerie = " + numeroSerie + "]";
    }

}
```

```java
public class Conductor {
    private String nombre;
    private String licencia;
    private Vehiculo vehiculo; // Asociación bidireccional

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

    public void setVehiculo(Vehiculo vehiculo) {
        this.vehiculo = vehiculo;
    }

    public Vehiculo getVehiculo() {
        return vehiculo;
    }

    @Override
    public String toString() {
        return "Conductor [nombre = " + nombre + ", licencia = " + licencia + "]";
    }
}
```

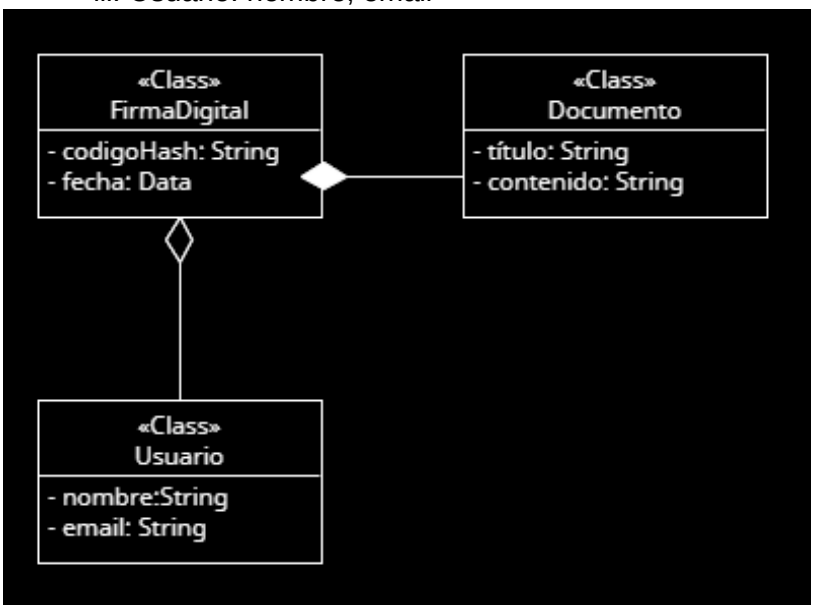
```java
public class Vehiculo {

    private String patente;
    private String modelo;
    private Motor motor;        // Agregación
    private Conductor conductor; // Asociación bidireccional

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

    public void setConductor(Conductor conductor) {
        this.conductor = conductor;
        conductor.setVehiculo(this); // Mantiene la bidireccionalidad
    }

    public Conductor getConductor() {
        return conductor;
    }

    public Motor getMotor() {
        return motor;
    }

    @Override
    public String toString() {
        return "Vehiculo [patente = " + patente + ", modelo = " + modelo + ", " +
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        // Crear motor
        Motor motor = new Motor("Diesel", "BB3Q-6006-FA");

        // Crear vehículo con motor (agregación)
        Vehiculo vehiculo = new Vehiculo("CC666CC", "Ford Ranger", motor);

        // Crear conductor
        Conductor conductor = new Conductor("Esteban Quito", "LIC-212706");

        // Asociar vehiculo ↔ conductor
        vehiculo.setConductor(conductor);

        // Mostrar resultados
        System.out.println(vehiculo);
        System.out.println("Conductor del vehículo: " + vehiculo.getConductor());
        System.out.println("Vehiculo del conductor: " + conductor.getVehiculo());
    }
}
```

```
ut
Debugger Console ×   TP5 (run) ×
run:
Vehiculo [patente = CC666CC, modelo = Ford Ranger, Motor [tipo = Diesel, numeroSerie = BB3Q-6006-FA]]
Conductor del vehículo: Conductor [nombre = Esteban Quito, licencia = LIC-212706]
Vehiculo del conductor: Vehiculo [patente = CC666CC, modelo = Ford Ranger, Motor [tipo = Diesel, numeroSerie =
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



```java
public class Usuario {

    private String nombre;
    private String email;

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

    public String getNombre() {
        return nombre;
    }

    public String getEmail() {
        return email;
    }

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

```java
public class FirmaDigital {

    private String codigoHash;
    private LocalDate fecha;
    private Usuario usuario; // Agregación

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

    public String getCodigoHash() {
        return codigoHash;
    }

    public LocalDate getFecha() {
        return fecha;
    }

    public Usuario getUsuario() {
        return usuario;
    }

    @Override
    public String toString() {
        return "FirmaDigital [codigoHash = " + codigoHash + ", fecha = " + fecha
    }
```

```java
public class Documento {

    private String titulo;
    private String contenido;
    private FirmaDigital firma; // Composición

    public Documento(String titulo, String contenido, FirmaDigital firma) {
        this.titulo = titulo;
        this.contenido = contenido;
        this.firma = firma;
    }

    public String getTitulo() {
        return titulo;
    }

    public String getContenido() {
        return contenido;
    }

    public FirmaDigital getFirma() {
        return firma;
    }

    @Override
    public String toString() {
        return "Documento [titulo = " + titulo + ", contenido = " + contenido + ",
    }
}
```

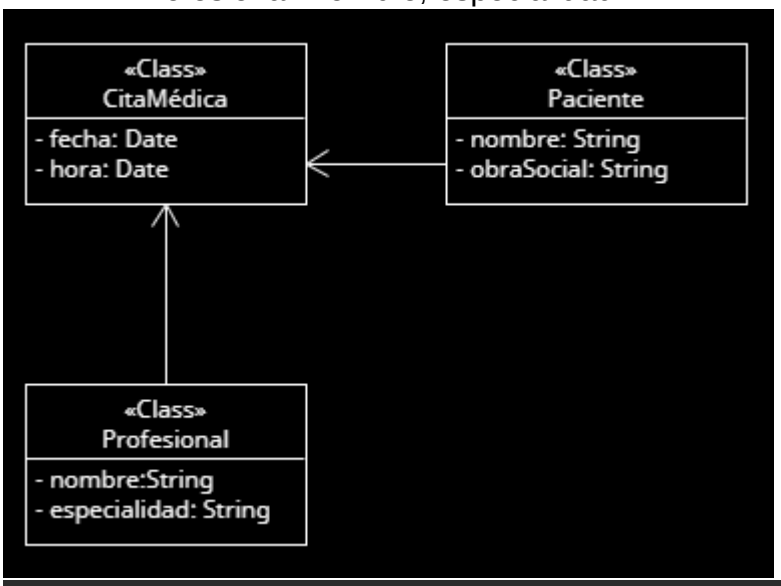
```java
public class Main {

    public static void main(String[] args) {
        // Crear usuario
        Usuario usuario = new Usuario("Vanina Araceli", "ara.vanina@email.com");

        // Crear firma digital (agregación con usuario)
        FirmaDigital firma = new FirmaDigital("FFF6V6V6", LocalDate.now(), usuario);

        // Crear documento con firma (composición)
        Documento doc = new Documento("Contrato", "Contenido del contrato...", firma);

        // Mostrar información
        System.out.println(doc);
        System.out.println("Usuario que firmó: \n" + doc.getFirma().getUsuario());
    }
}
```

```
ut
Debugger Console ×   TP5 (run) ×
run:
Documento [titulo = Contrato, contenido = Contenido del contrato..., firma = FirmaDigital [codigoHash = FFF
Usuario que firmó:
Usuario [nombre = Vanina Araceli, email = ara.vanina@email.com]
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




```java
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 String getObraSocial() {
        return obraSocial;
    }

    @Override
    public String toString() {
        return "Paciente [nombre = " + nombre + ", 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 String getEspecialidad() {
        return especialidad;
    }

    @Override
    public String toString() {
        return "Profesional [nombre = " + nombre + ", especialidad = " + especia
    }
}
```

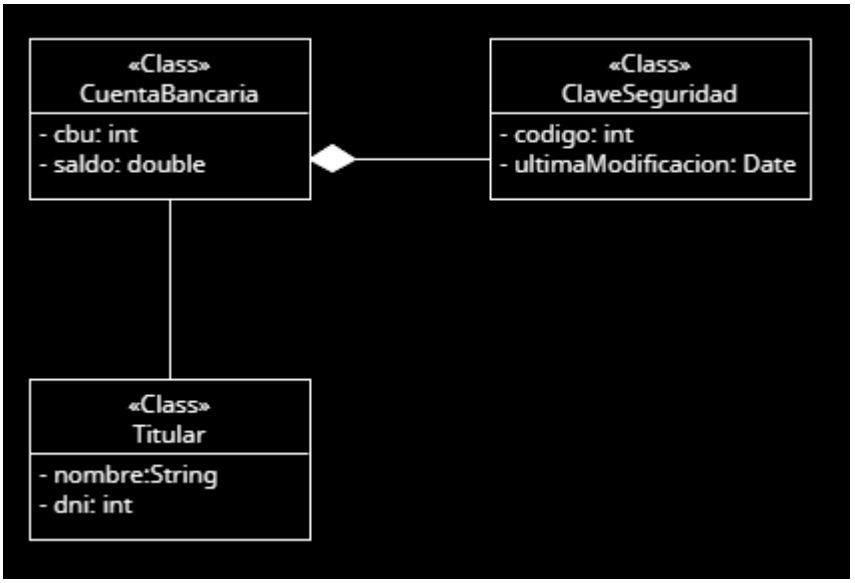
```java
public class CitaMedica {

    private LocalDate fecha;
    private String hora;
    private Paciente paciente;       // Asociación unidireccional
    private Profesional profesional; // Asociación unidireccional

    public CitaMedica(LocalDate fecha, String hora, Paciente paciente, Profesional p
        this.fecha = fecha;
        this.hora = hora;
        this.paciente = paciente;
        this.profesional = profesional;
    }

    public LocalDate getFecha() {
        return fecha;
    }

    public String getHora() {
        return hora;
    }

    public Paciente getPaciente() {
        return paciente;
    }

    public Profesional getProfesional() {
        return profesional;
    }

    @Override
    public String toString() {
        return "CitaMédica [fecha = " + fecha + ", hora = " + hora + ", paciente = "
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        // Crear paciente
        Paciente paciente = new Paciente("Flavio Gutierrez", "Medife");

        // Crear profesional
        Profesional profesional = new Profesional("Dra. Meli", "Cardiología");

        // Crear cita médica
        CitaMedica cita = new CitaMedica(LocalDate.of(2025, 11, 7), "10:30", paciente, p

        // Mostrar datos
        System.out.println(cita);
        System.out.println("Paciente: " + cita.getPaciente().getNombre());
        System.out.println("Profesional: " + cita.getProfesional().getNombre());
    }
}
```

```
Debugger Console ×   TP5 (run) ×
run:
CitaMédica [fecha = 2025-11-07, hora = 10:30, paciente = Paciente [nombre = Flavio Gutierrez, obraSoci
Paciente: Flavio Gutierrez
Profesional: Dra. Meli
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.



```java
public class ClaveSeguridad {

    private int codigo;
    private LocalDate ultimaModificacion;

    public ClaveSeguridad(int codigo, LocalDate ultimaModificacion) {
        this.codigo = codigo;
        this.ultimaModificacion = ultimaModificacion;
    }

    public int getCodigo() {
        return codigo;
    }

    public LocalDate getUltimaModificacion() {
        return ultimaModificacion;
    }

    @Override
    public String toString() {
        return "ClaveSeguridad [codigo = " + codigo + ", ultimaModificacion =
    }
}
```

```java
public class Titular {

    private String nombre;
    private int dni;
    private CuentaBancaria cuenta; // Asociación bidireccional

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

    public void setCuenta(CuentaBancaria cuenta) {
        this.cuenta = cuenta;
    }

    public CuentaBancaria getCuenta() {
        return cuenta;
    }

    @Override
    public String toString() {
        return "Titular [nombre = " + nombre + ", dni = " + dni + "]";
    }
}
```

```java
public class CuentaBancaria {

    private String cbu;
    private double saldo;
    private ClaveSeguridad clave;  // Composición
    private Titular titular;        // Asociación bidireccional

    public CuentaBancaria(String cbu, double saldo, ClaveSeguridad clave) {
        this.cbu = cbu;
        this.saldo = saldo;
        this.clave = clave;
    }

    public void setTitular(Titular titular) {
        this.titular = titular;
        titular.setCuenta(this); // Mantener bidireccionalidad
    }

    public Titular getTitular() {
        return titular;
    }

    public ClaveSeguridad getClave() {
        return clave;
    }

    @Override
    public String toString() {
        return "CuentaBancaria [CBU = " + cbu + ", saldo = " + saldo + ", clave
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        // Crear clave de seguridad
        ClaveSeguridad clave = new ClaveSeguridad(2127, LocalDate.now());

        // Crear cuenta bancaria (con composición hacia clave)
        CuentaBancaria cuenta = new CuentaBancaria("725-002330", 95000.0, clave);

        // Crear titular
        Titular titular = new Titular("Mayra Diaz", 45000208);

        // Asociar titular y cuenta (bidireccional)
        cuenta.setTitular(titular);

        // Mostrar resultados
        System.out.println(cuenta);
        System.out.println("Titular de la cuenta: " + cuenta.getTitular());
        System.out.println("Cuenta del titular: " + titular.getCuenta());
    }
```

```
ut
Debugger Console ×   TP5 (run) ×
run:
CuentaBancaria [CBU = 725-002330, saldo = 95000.0, clave = ClaveSeguridad [codigo = 2127, ultimaModificacion =
Titular de la cuenta: Titular [nombre = Mayra Diaz, dni = 45000208]
Cuenta del titular: CuentaBancaria [CBU = 725-002330, saldo = 95000.0, clave = ClaveSeguridad [codigo = 2127, u
BUILD SUCCESSFUL (total time: 0 seconds)
```

DEPENDENCIA DE USO

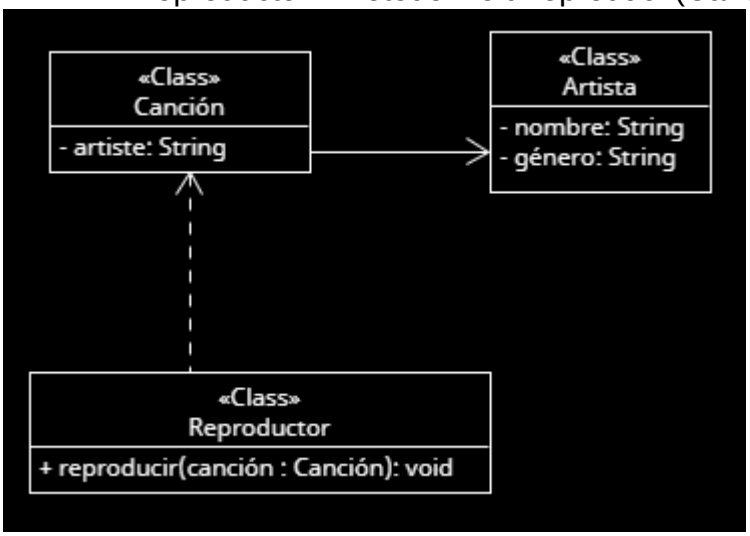
La clase usa otra como parámetro de un método, pero no la guarda como atributo.

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)



```java
public class Artista {

    private String nombre;
    private String genero;

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

    public String getNombre() {
        return nombre;
    }

    public String getGenero() {
        return genero;
    }

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

```java
public class Cancion {

    private String titulo;
    private Artista artista; // Asociación unidireccional

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

    public String getTitulo() {
        return titulo;
    }

    public Artista getArtista() {
        return artista;
    }

    @Override
    public String toString() {
        return "Canción [título=" + titulo + ", artista=" + artista.getNombre()
    }
}
```

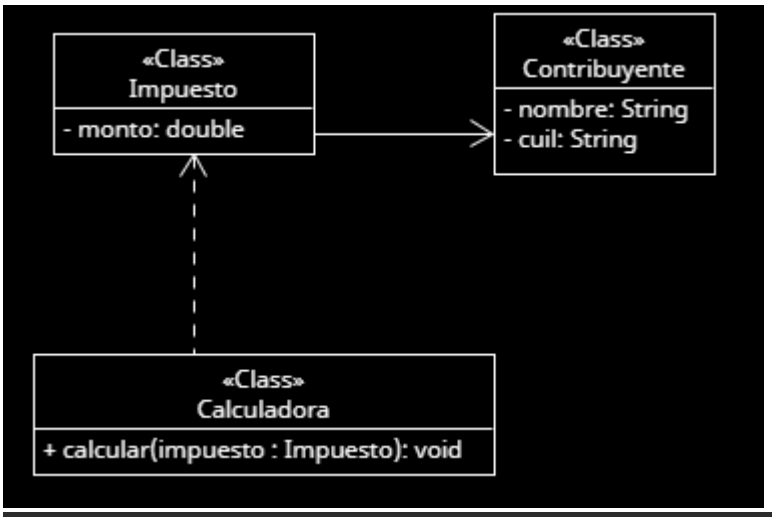
```java
public class Reproductor {

    // Dependencia de uso: el objeto Cancion se recibe como parámetro,
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciiendo: " + cancion.getTitulo() +
                " - " + cancion.getArtista().getNombre());
    }
}
```

```java
public class Main {

    public static void main(String[] args) {

        // Crear artista
        Artista artista = new Artista("The Killers", "Rock Inde");

        // Crear canción (asociación con artista)
        Cancion cancion = new Cancion("Mr. Brightside", artista);

        // Crear reproductor y usar dependencia de uso
        Reproductor reproductor = new Reproductor();
        reproductor.reproducir(cancion);
    }
}
```

```
ut - TP5 (run)
run:
Reproduciiendo: Mr. Brightside - The Killers
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)




```java
public class Contribuyente {

    private String nombre;
    private String cuil;

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

    public String getNombre() {
        return nombre;
    }

    public String getCuil() {
        return cuil;
    }

    @Override
    public String toString() {
        return "Contribuyente [nombre=" + nombre + ", CUIL=" + cuil + "]";
    }
}
```

```java
public class Impuesto {

    private double monto;
    private Contribuyente contribuyente; // Asociación unidireccional

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

    public double getMonto() {
        return monto;
    }

    public Contribuyente getContribuyente() {
        return contribuyente;
    }

    @Override
    public String toString() {
        return "Impuesto [monto=" + monto + ", contribuyente=" + contribu
    }
}
```

```java
public class Calculadora {

    // Dependencia de uso: usa Impuesto como parámetro pero no lo guarda
    public void calcular(Impuesto impuesto) {
        double monto = impuesto.getMonto();
        double total = monto + (monto * 0.21); // Ejemplo: aplica 21% de IVA

        System.out.println("Calculando impuesto para: " + impuesto.getContribuye
        System.out.println("Monto base: $" + monto);
        System.out.println("Monot total con IVA: $" + total);
    }
}
```

```java
public class Main {

    public static void main(String[] args) {

        // Crear Contribuyente
        Contribuyente contribuyente = new Contribuyente("Luz Montero", "20-98211198-3");

        // Crear impuesto (asociación unidireccional con contribuyente)
        Impuesto impuesto = new Impuesto(60000.0, contribuyente);

        // Crear calculadora (dependencia de uso)
        Calculadora calc = new Calculadora();
        calc.calcular(impuesto);
    }
}
```

```
ut - TP5 (run) ×
run:
Calculando impuesto para: Luz Montero
Monto base: $60000.0
Monot total con IVA: $72600.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

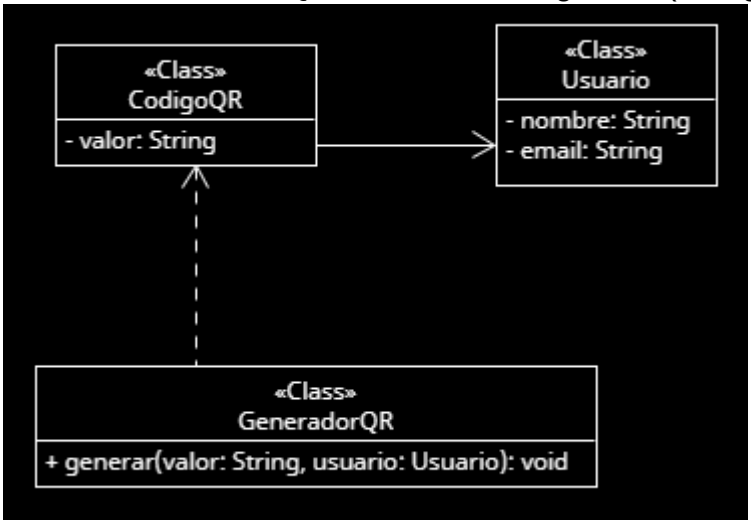
DEPENDENCIA DE CREACIÓN

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)



```java
public class Usuario {

    private String nombre;
    private String email;

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

    public String getNombre() {
        return nombre;
    }

    public String getEmail() {
        return email;
    }

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

```java
public class CodigoQR {

    private String valor;
    private Usuario usuario; // Asociación unidireccional

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

    public String getValor() {
        return valor;
    }

    public Usuario getUsuario() {
        return usuario;
    }

    @Override
    public String toString() {
        return "CódigoQR [valor = " + valor + ", usuario = " + usua
    }
}
```

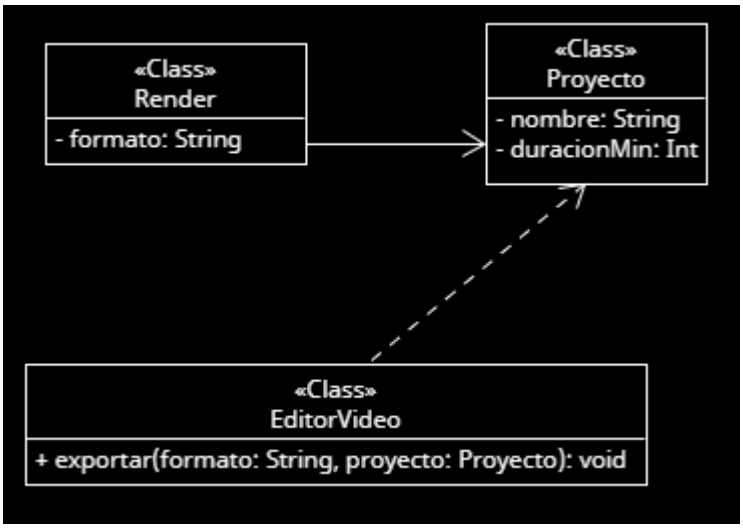
```java
public class GeneradorQR {

    // Dependencia de creación: crea un objeto de otra clase, pe
    public void generar(String valor, Usuario usuario) {
        CodigoQR qr = new CodigoQR(valor, usuario);
        System.out.println("Generando código QR...");
        System.out.println(qr);
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        // Crear usuario
        Usuario usuario = new Usuario("Ailin Ando", "ailin.a@email.com");

        // Crear generador y generar código QR (dependencia de creación)
        GeneradorQR generador = new GeneradorQR();
        generador.generar("70D4P4T1", usuario);
    }
}
```

```
ercicio13.Main > main >
t - TP5 (run) ×
run:
Generando código QR...
CódigoQR [valor = 70D4P4T1, usuario = Ailin Ando]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Clases y atributos:
    i. Render: formato.
    ii. Proyecto: nombre, duracionMin.
    iii. EditorVideo->método: void exportar(String formato, Proyecto proyecto)




```java
public class Proyecto {

    private String nombre;
    private int duracionMin;

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

    public String getNombre() {
        return nombre;
    }

    public int getDuracionMin() {
        return duracionMin;
    }

    @Override
    public String toString() {
        return "Proyecto [nombre = " + nombre + ", duración = " +
    }
}
```

```java
public class Render {

    private String formato;
    private Proyecto proyecto;

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

    public String getFormato() {
        return formato;
    }

    public Proyecto getProyecto() {
        return proyecto;
    }

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

```java
public class EditorVideo {
    public void exportar(String formato, Proyecto proyecto) {

        // Dependencia de creación: se crea un Render dentro del método
        Render render = new Render(formato, proyecto);
        System.out.println("Exportando render...");
        System.out.println(render);
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
        Proyecto proyecto = new Proyecto("I, Pet Goat II", 450);
        EditorVideo editor = new EditorVideo();
        editor.exportar("MP4", proyecto);
    }
}
```

```
ut - TP5 (run)
run:
Exportando render...
Render [formato = MP4, proyecto = I, Pet Goat II]
BUILD SUCCESSFUL (total time: 0 seconds)
```