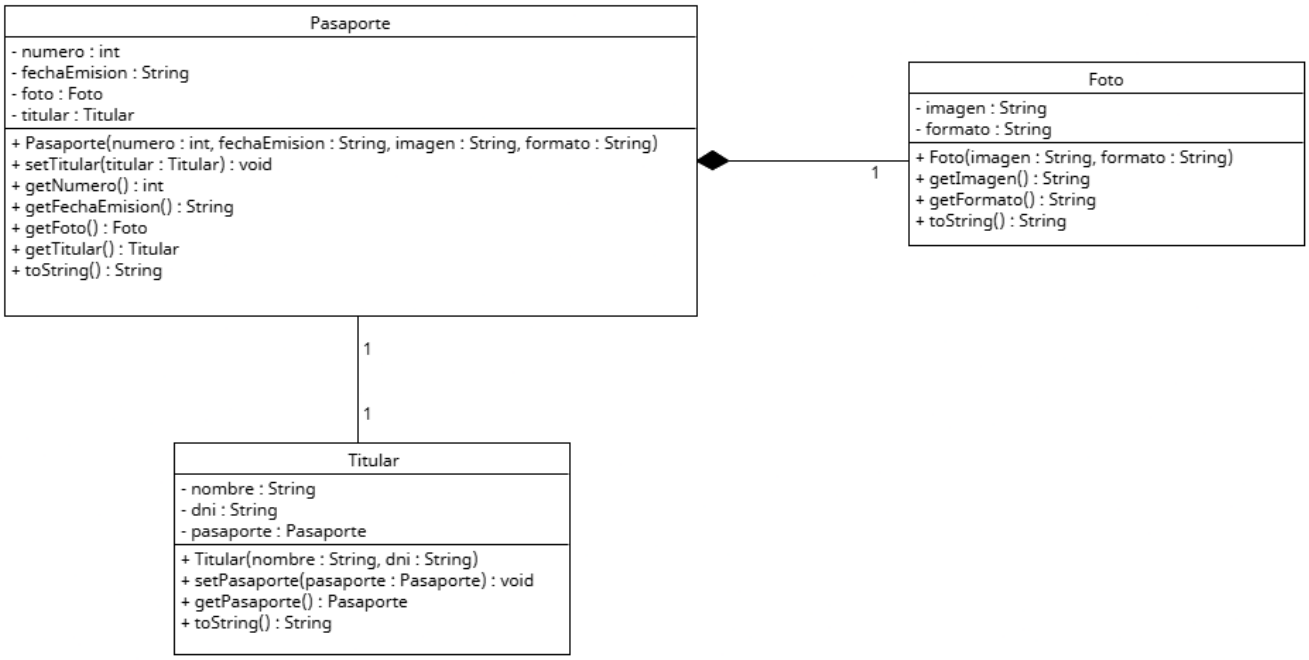
PROGRAMACION II

Trabajo practico 5: Relaciones UML 1 a 1.

Repositorio Github: https://github.com/AguP10/UTN-TUPaD-P2.git

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

Diagrama



Tipo de relación

- Pasaporte -> Foto: Composición. Unidireccional.
- Pasaporte -> Titular: Asociación. Bidireccional.

Código:

Clase Pasaporte

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

    public Pasaporte(int numero, String fechaEmision, String imagen, String formato) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(imagen, formato);
    }
    public void setTitular(Titular titular) {
        this.titular = titular;
        if (titular != null && titular.getPasaporte() != this) {
            titular.setPasaporte(this);
        }
    }
    public int 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 + ", fechaEmision=" + fechaEmision
                + ", foto=" + foto + ", titular=" + titular + '}';
    }
}
```

Clase Foto

```
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=" + imagen + ", formato=" + formato + '}';
    }
}
```

Clase Titular

```
public 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;
        if (pasaporte != null && pasaporte.getTitular() != this) {
```

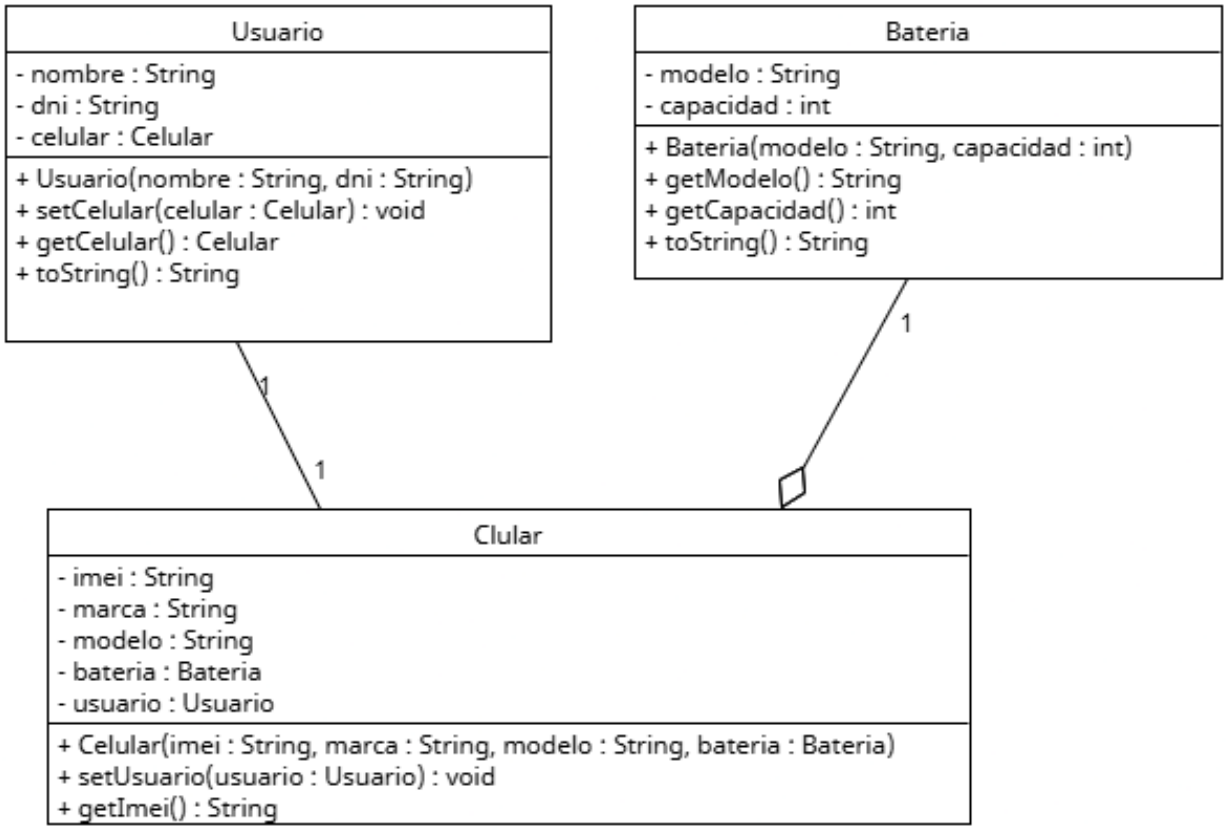
```
            pasaporte.setTitular(this);
        }
    }

    public Pasaporte getPasaporte() {
        return pasaporte;
    }

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

Main

```
public class TP5 {

    public static void main(String[] args) {
        Titular titular = new Titular("Ernesto Diaz", "12345678");
        Pasaporte pasaporte = new Pasaporte(53132, "24-09-25", "Foto.jpg", "JPEG");

        pasaporte.setTitular(titular);

        System.out.println(pasaporte);
        System.out.println(titular.getPasaporte());
    }
}
```

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

Diagrama



Tipo de relación

- Celular -> batería: Agregación. Unidireccional.
- Celular -> usuario: Asociación. Bidireccional.

Código:

Clase Batería

```
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 + '}';
        }
}
```

Clase Usuario

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

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

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

```
    public Celular getCelular() {
        return celular;
    }

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

Calase Celular

```
public class Celular {
    private String imei;
    private String marca;
    private String modelo;
    private Bateria bateria;
    private Usuario usuario;

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

```
    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;
    }

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

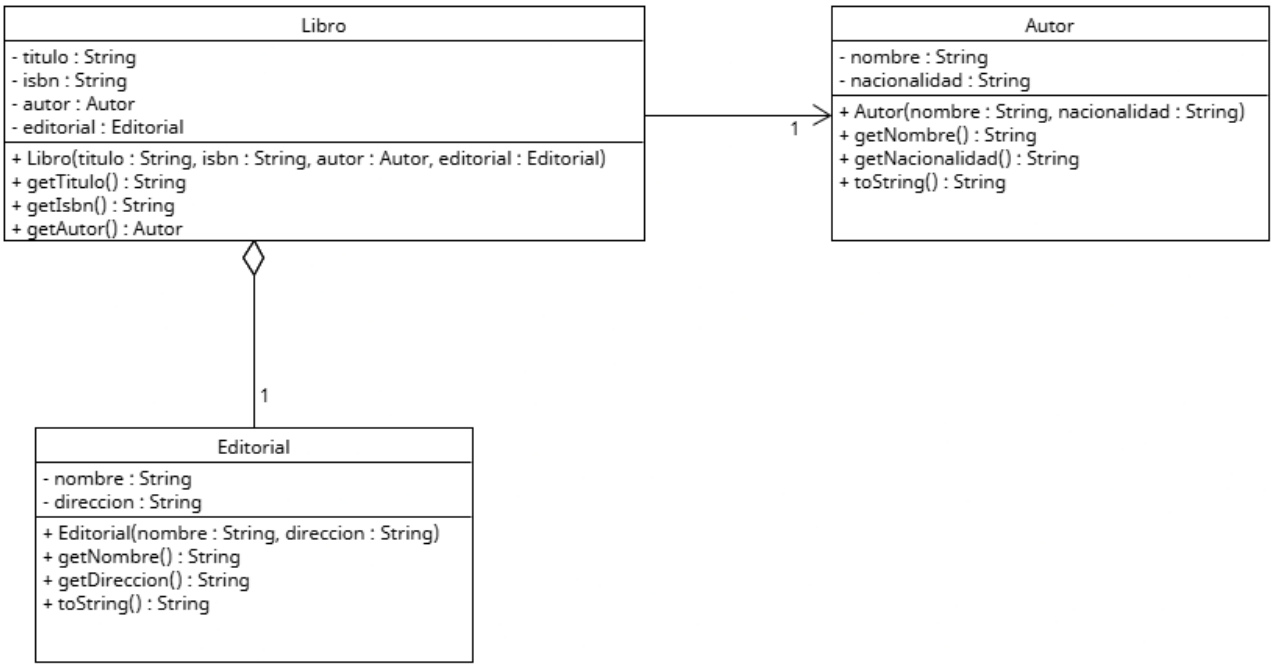
Main

```
public class TP5 {
    public static void main(String[] args) {
        Bateria bateria = new Bateria("Samsung BG965ABU", 4000);
```

```
        Celular celular = new Celular("123456789", "Samsung", "Galaxy S25", bateria);
        Usuario usuario = new Usuario("Juan Perez", "36654321");


        celular.setUsuario(usuario);
        System.out.println(celular);
        System.out.println(usuario.getCelular());
    }
}
```

3. **Libro - Autor – Editorial**
    a. **Asociación unidireccional: Libro → Autor**
    b. **Agregación: Libro → Editorial**

Diagrama



Tipo de relación

- Libro -> Autor: Asociación. Unidireccional.
- Libro -> Editorial: Agregación. Unidireccional.

Código

Clase Autor

```
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 + '}';
    }
}
```

Clase Editorial

```
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 + '}';
    }
}
```

Clase Libro

```
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;
    }

    public Editorial getEditorial() {
        return editorial;
    }

    @Override
    public String toString() {
        return "Libro{" + "titulo=" + titulo + ", isbn=" + isbn +
                ", autor=" + autor + ", editorial=" + editorial + '}';
    }
}
```

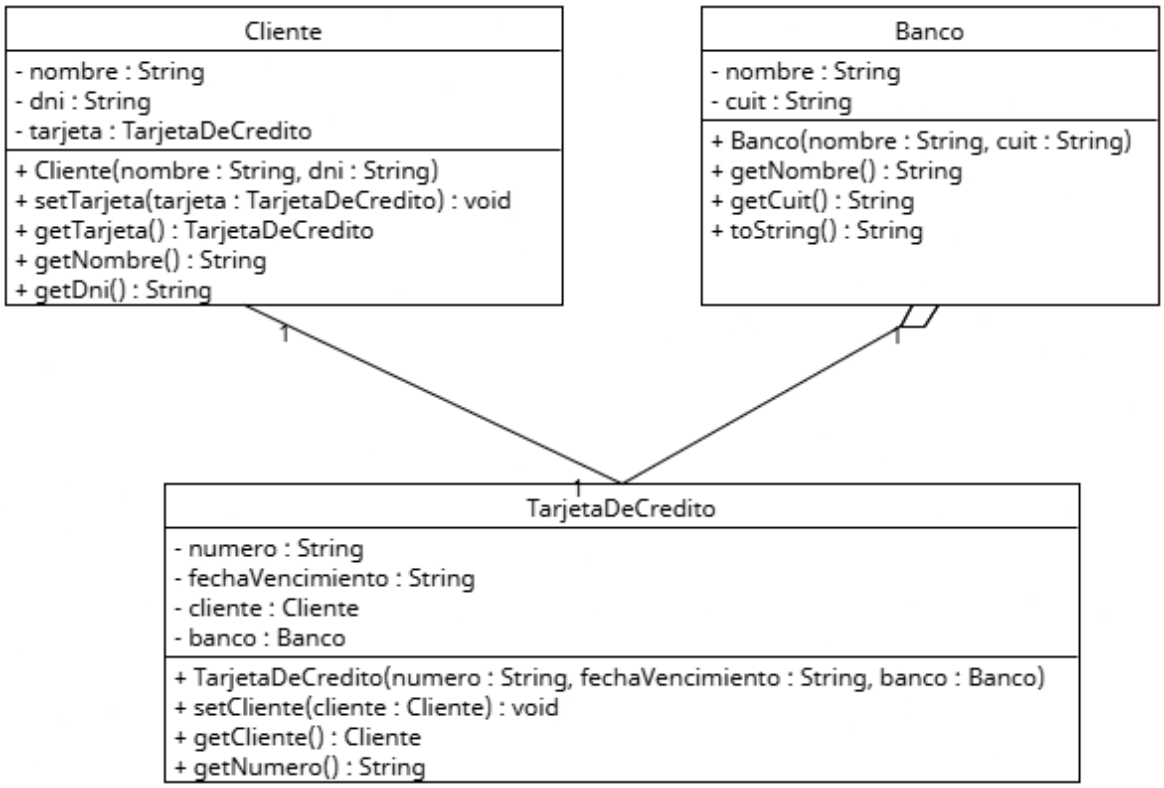
Main

```
public class TP5 {
    public static void main(String[] args) {
        Autor autor = new Autor("Gabriel García Márquez", "Colombiana");
        Editorial editorial = new Editorial("Codice", "Rivadavia 125");
        Libro libro = new Libro("Cien años de soledad", "54612318", autor, editorial);

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

```
    }
}
```

4. **TarjetaDeCrédito - Cliente - Banco**
    a. **Asociación bidireccional: TarjetaDeCrédito ↔ Cliente**
    b. **Agregación: TarjetaDeCrédito → Banco**

Diagrama



Tipo de relaciones

- TarjetaDeCredito <-> Cliente: Asociación. Bidireccional.
- TarjetaDeCredito -> Banco: Agregación. Unidireccional.

Código

Clase Cliente

```
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;
        if (tarjeta != null && tarjeta.getCliente() != this) {
            tarjeta.setCliente(this);
        }
    }

    public TarjetaDeCredito getTarjeta() {
        return tarjeta;
    }

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

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

Clase Banco

```
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;
    }

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

Clase TarjetaDeCredito

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

    public TarjetaDeCredito(String numero, String fechaVencimiento, Banco banco) {
```

```
        this.numero = numero;
        this.fechaVencimiento = fechaVencimiento;
        this.banco = banco;
    }

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

    public Cliente getCliente() {
        return cliente;
    }

    public String getNumero() {
        return numero;
    }

    public String getFechaVencimiento() {
        return fechaVencimiento;
    }

    public Banco getBanco() {
        return banco;
    }
      @Override
        public String toString() {
            return "TarjetaDeCredito{" + "numero=" + numero + ", fechaVencimiento=" + fechaVencimiento +
", cliente=" + cliente + ", banco=" + banco + '}';
        }
```

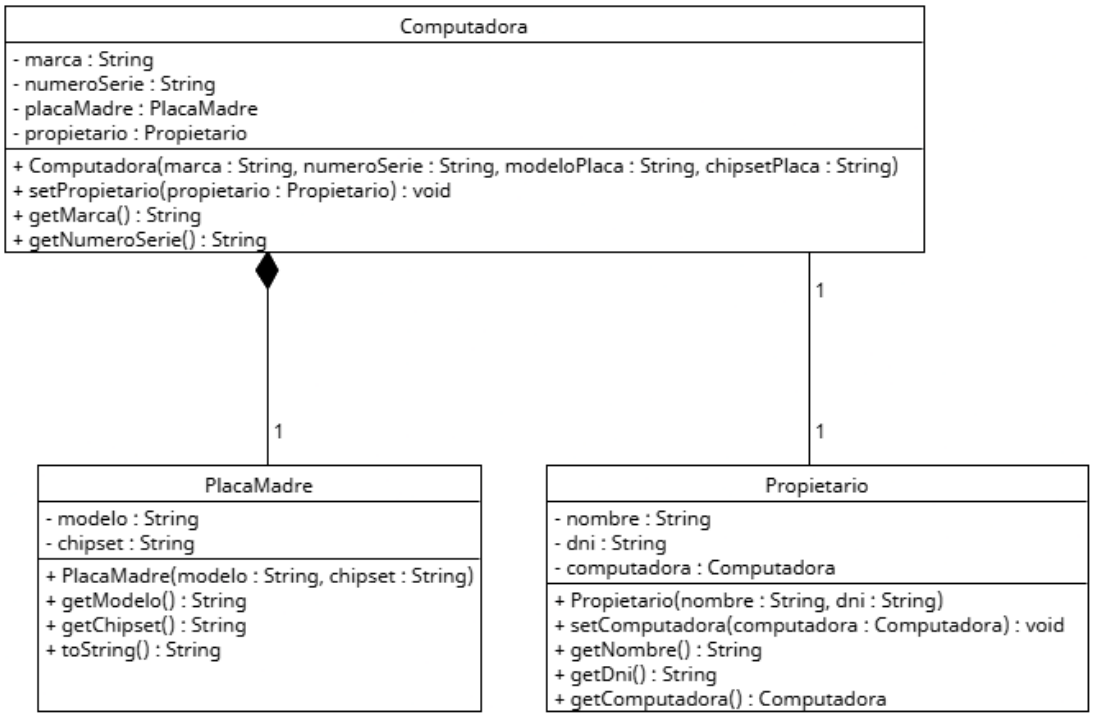
```
}
```

Main

```
public class TP5 {
    public static void main(String[] args) {
        Banco banco = new Banco("Banco Santander", "20-123456-9");
        TarjetaDeCredito tarjeta = new TarjetaDeCredito("1234-5678-9012-3456", "10/30", banco);
        Cliente cliente = new Cliente("Lucia Gomez", "12345678");

        tarjeta.setCliente(cliente);

        System.out.println(tarjeta);
        System.out.println(cliente.getTarjeta());
    }
}
```

5. **Computadora - PlacaMadre - Propietario**
    a. **Composición: Computadora → PlacaMadre**
    b. **Asociación bidireccional: Computadora ↔ Propietario**

Diagrama



Tipo de relación

- Computadora -> PlacaMadre: Composición. Unidireccional.
- Computadora <-> PlacaMadre: Asociación. Bidireccional.

Código

Clase Propietario

```
public class Propietario {
    private String nombre;
    private String dni;
    private Computadora computadora;

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

    public String getNombre() {
        return nombre;
    }

    public String getDni() {
        return dni;
    }

    public Computadora getComputadora() {
        return computadora;
    }

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

Clase PlacaMadre

```
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;
    }
}
```

// Clase Computadora

```
public class Computadora {
    private String marca;
    private String numeroSerie;
    private PlacaMadre placaMadre;
    private Propietario propietario;

    public Computadora(String marca, String numeroSerie, String modeloPlaca, String chipsetPlaca) {
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        this.placaMadre = new PlacaMadre(modeloPlaca, chipsetPlaca);
```

```
    }

    public String getMarca() {
        return marca;
    }

    public String getNumeroSerie() {
        return numeroSerie;
    }

    public PlacaMadre getPlacaMadre() {
        return placaMadre;
    }

    public Propietario getPropietario() {
        return propietario;
    }

    public void setPropietario(Propietario propietario) {
        this.propietario = propietario;
        propietario.setComputadora(this);
    }
}
```

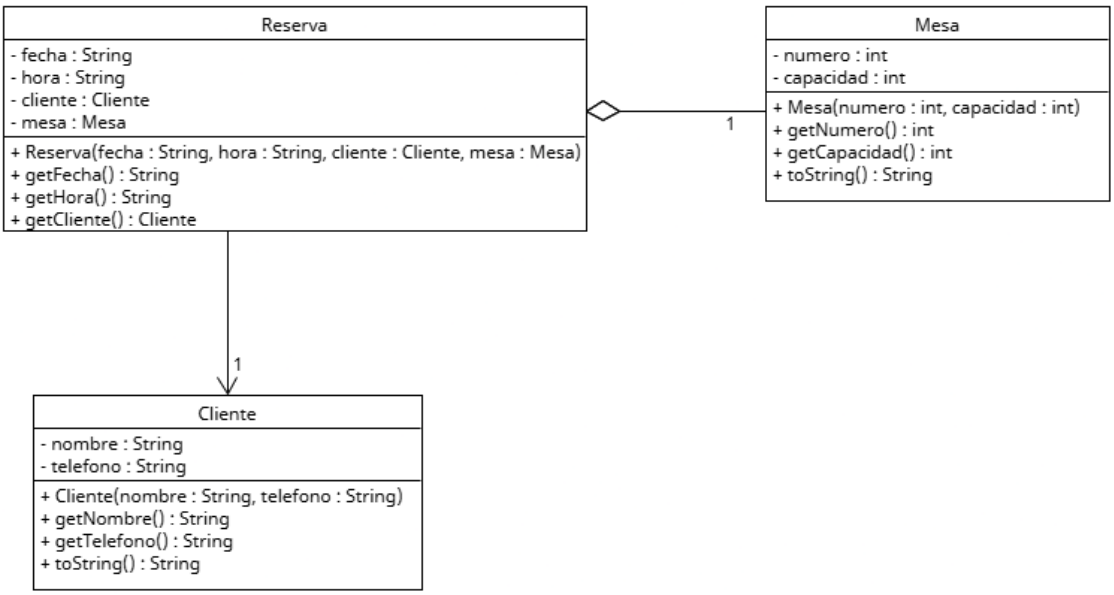
Main

```
public class TP5 {
    public static void main(String[] args) {
        Propietario propietario = new Propietario("Pedro Gomez", "4521354");
        Computadora pc = new Computadora("ASUS", "SN12345", "ASUS X870-PLUS", "870");
        pc.setPropietario(propietario);
        System.out.println("Propietario: " + propietario.getNombre());
        System.out.println("Computadora: " + propietario.getComputadora().getMarca());
```

```
        System.out.println("Placa Madre: " + pc.getPlacaMadre().getModelo());
    }
}
```

6. **Reserva - Cliente - Mesa**
    a. **Asociación unidireccional: Reserva → Cliente**
    b. **Agregación: Reserva → Mesa**

Diagrama



Tipo de relación

- Reserva -> Cliente: Asociación. Unidireccional.
- Reserva -> Mesa: Agregación. Unidireccional.

Código

Clase Cliente

```
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 + ", telefono=" + telefono + '}';
    }
}
```

Clase Mesa

```
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 + '}';
    }
}
```

Clase Reserva

```
public class Reserva {
    private String fecha;
    private String hora;
    private Cliente cliente;
    private Mesa mesa;

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

     public String getFecha() {
        return fecha;
    }

    public String getHora() {
        return hora;
    }

    public Cliente getCliente() {
        return cliente;
    }
```

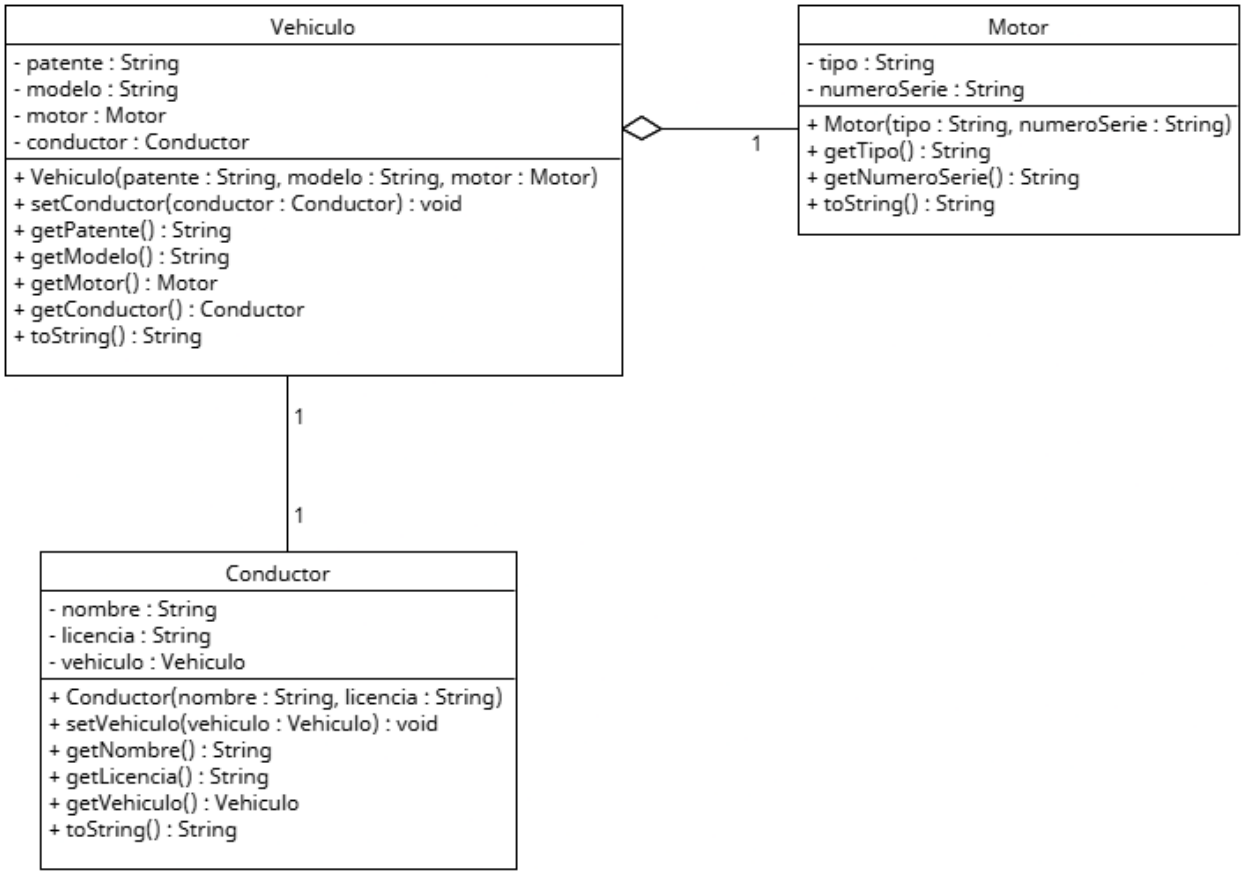
```
    public Mesa getMesa() {
        return mesa;
    }

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

Main

```
public class TP5 {
    public static void main(String[] args) {
        Cliente cliente = new Cliente("Ana López", "123456789");
        Mesa mesa = new Mesa(5, 4);

        Reserva reserva = new Reserva("2025-09-26", "20:00", cliente, mesa);

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

7. **Vehículo - Motor – Conductor**
    a. **Agregación: Vehículo → Motor**
    b. **Asociación bidireccional: Vehículo ↔ Conductor**

Diagrama



Tipo de relación

- Vehículo -> Motor: Agregación. Unidireccional.
- Vehículo -> Conductor: Asociación. Bidireccional.

Código

Clase Conductor

```
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.getConductor() != this) {
            vehiculo.setConductor(this);
        }
    }

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

Clase Motor

```
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 + '}';
    }
}
```

Clase Vehículo

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

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

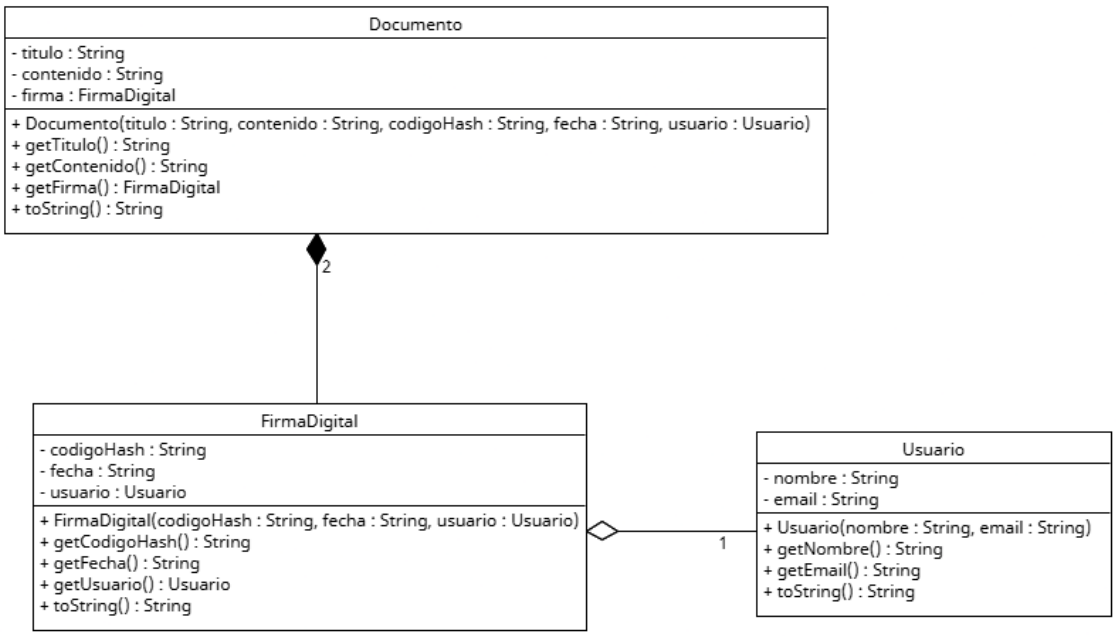
```
    public String getPatente() {
        return patente;
    }

    public String getModelo() {
        return modelo;
    }

    public Motor getMotor() {
        return motor;
    }

    public Conductor getConductor() {
        return conductor;
    }

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

    @Override
    public String toString() {
        return "Vehiculo{" +
                "patente='" + patente + '\'' +  ", modelo='" + modelo + '\'' + ", motor=" + motor + ", conductor="
+ conductor + '}';
    }
}
```

Main

```
public class TP5 {

    public static void main(String[] args) {

        Motor motor = new Motor("Diesel", "H542");

        Vehiculo vehiculo = new Vehiculo("AF251DD", "Ford Ranger", motor);

        Conductor conductor = new Conductor("Carlos Gómez", "L1235");


        vehiculo.setConductor(conductor);


        System.out.println(vehiculo);

        System.out.println(conductor.getVehiculo().getPatente());

    }

}
```

8. **Documento - FirmaDigital - Usuario**
    a. **Composición: Documento → FirmaDigital**
    b. **Agregación: FirmaDigital → Usuario**

Diagrama



Tipo de relación

- Documento -> FirmaDigital: Composición. Unidireccional.
- FirmaDigital -> Usuario: Agregación. Unidireccional.

Código

Clase 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;
    }

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

Clase FirmaDigital

```
public class FirmaDigital {
    private String codigoHash;
    private String fecha;
    private Usuario usuario;
```

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

    public String getCodigoHash() {
        return codigoHash;
    }

    public String getFecha() {
        return fecha;
    }

    public Usuario getUsuario() {
        return usuario;
    }

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

Clase Documento

```
public class Documento {
    private String titulo;
    private String contenido;
    private FirmaDigital firma;
```

```
    public Documento(String titulo, String contenido, String codigoHash, String fecha, Usuario usuario) {
        this.titulo = titulo;
        this.contenido = contenido;
         this.firma = new FirmaDigital(codigoHash, fecha, usuario);
    }

    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 + '\'' +
                ", firma=" + firma +
                '}';
    }
}
```

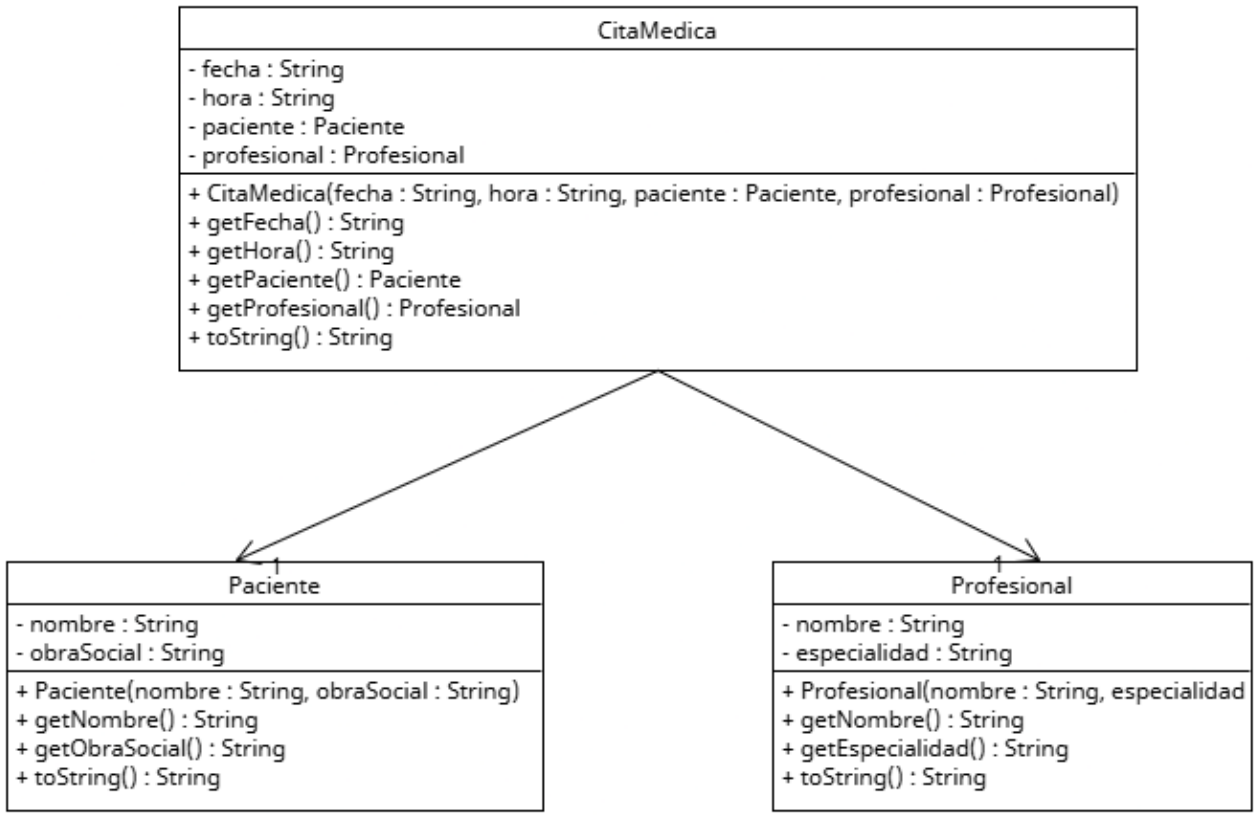
Main

```
public class TP5 {
    public static void main(String[] args) {
        Usuario usuario = new Usuario("Laura Pérez", "laura@mail.com");
```

```
    Documento doc = new Documento("Contrato", "Contenido del contrato", " 9c8245e6e", "2025-09-25", usuario);


    System.out.println(doc);

    System.out.println("Usuario de la firma: " + doc.getFirma().getUsuario().getNombre());

  }

}
```

9. **CitaMédica - Paciente - Profesional**
    a. **Asociación unidireccional: CitaMédica → Paciente**
    b. **Asociación unidirecciona: CitaMédica → Profesional**

Diagrama



Tipo de relación

- CitaMedica -> Paciente: Asociación. Unidireccional.
- CitaMedica -> Profesional. Asociación. Unidireccional.

Código

Clase Paciente

```
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 + '}';
    }
}
```

Clase Profesional

```
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=" + especialidad + '}';
    }
}
```

Clase CitaMedica

```
public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente;
    private Profesional profesional;

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

    public String getFecha() {
```

```
    return fecha;
  }

  public String getHora() {
    return hora;
  }

  public Paciente getPaciente() {
    return paciente;
  }

  public Profesional getProfesional() {
    return profesional;
  }

  @Override
  public String toString() {
    return "CitaMedica{" +
        "fecha='" + fecha + '\'' + ", hora='" + hora + '\'' + ", paciente=" + paciente +
        ", profesional=" + profesional +
        '}';
  }
}
```

Main

```
public class Main {
  public static void main(String[] args) {
    Paciente paciente = new Paciente("Juan Pérez", "OSDE");
    Profesional profesional = new Profesional("Dra. Andrea López", "Cardiología");

    CitaMedica cita = new CitaMedica("2025-09-26", "10:30", paciente, profesional);
```

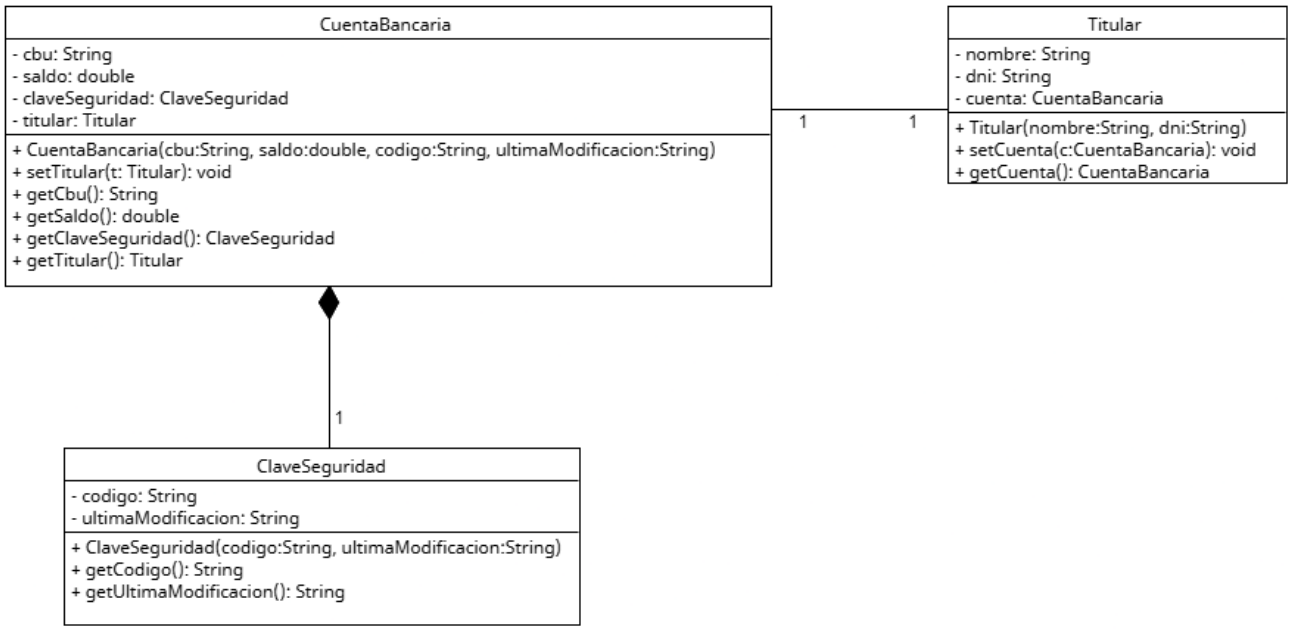
```
        System.out.println(cita);
        System.out.println("Paciente de la cita: " + cita.getPaciente().getNombre());
        System.out.println("Profesional de la cita: " + cita.getProfesional().getNombre());
    }
}
```

10. CuentaBancaria - ClaveSeguridad - Titular

a. **Composición: CuentaBancaria → ClaveSeguridad**

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

Diagrama



Tipo de relación

- CuentaBancaria -> ClaveSeguridad: Composición. Bidireccional.
- CuentaBancaria <-> Titular. Asociación. Bidireccional.

Código

Clase Cuenta Bancaria

```
public class CuentaBancaria {
    private String cbu;
    private double saldo;
```

```
private ClaveSeguridad claveSeguridad;
private Titular titular;

public CuentaBancaria(String cbu, double saldo, String codigo, String ultimaModificacion) {
    this.cbu = cbu;
    this.saldo = saldo;
    this.claveSeguridad = new ClaveSeguridad(codigo, ultimaModificacion);
}

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

public String getCbu() {
    return cbu;
}

public double getSaldo() {
    return saldo;
}

public ClaveSeguridad getClaveSeguridad() {
    return claveSeguridad;
}

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

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

Clase ClaveSeguridad

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

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

    public String getCodigo() {
        return codigo;
    }

    public String getUltimaModificacion() {
        return ultimaModificacion;
    }

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

Clase Titular

```
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;
        if (cuenta != null && cuenta.getTitular() != this) {
            cuenta.setTitular(this);
        }
    }

    public CuentaBancaria getCuenta() {
        return cuenta;
    }

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

Main

```
public class TP5 {
    public static void main(String[] args) {
        Titular titular = new Titular("Raul Perez", "123456321");
```

```
        CuentaBancaria cuenta = new CuentaBancaria("213215621324", 25000.50, "ABC123", "25-09-2025");


        cuenta.setTitular(titular);


        System.out.println(cuenta);
        System.out.println(titular.getCuenta());
    }
}
```

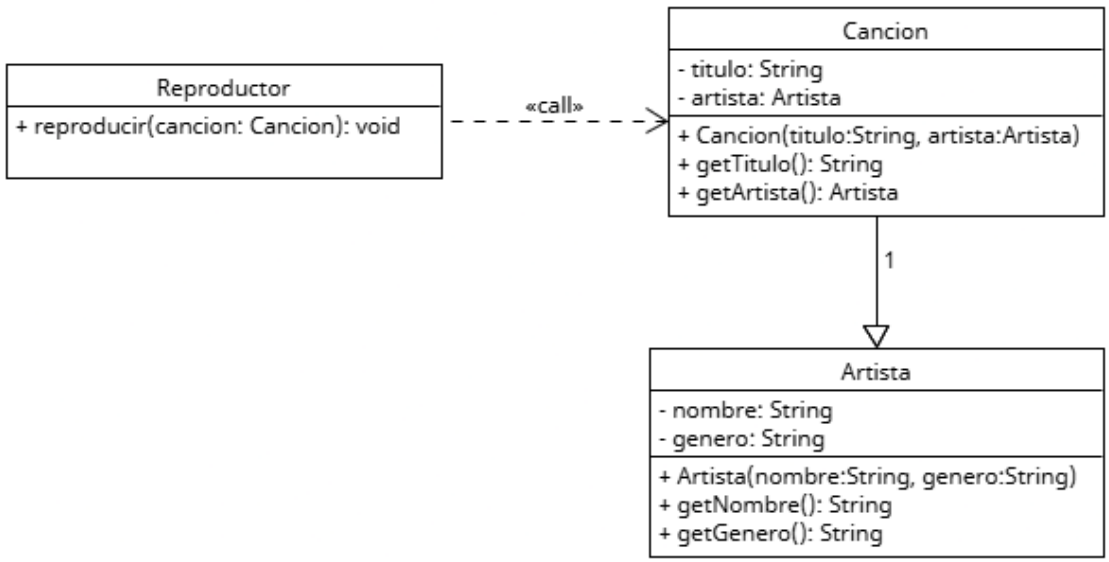
Dependencia de uso

11. Reproductor - Canción - Artista

a. Asociación unidireccional: Canción → Artista

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

Diagrama



Código

Clase 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 String getGenero() {
        return genero;
    }

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

Clase Canción

```
public class Cancion {
    private String titulo;
    private Artista artista;

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

Clase Reproductor

```
public class Reproductor {
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciendo: " + cancion.getTitulo() +
                " - " + cancion.getArtista().getNombre() +
                " (" + cancion.getArtista().getGenero() + ")");
    }
}
```

Main

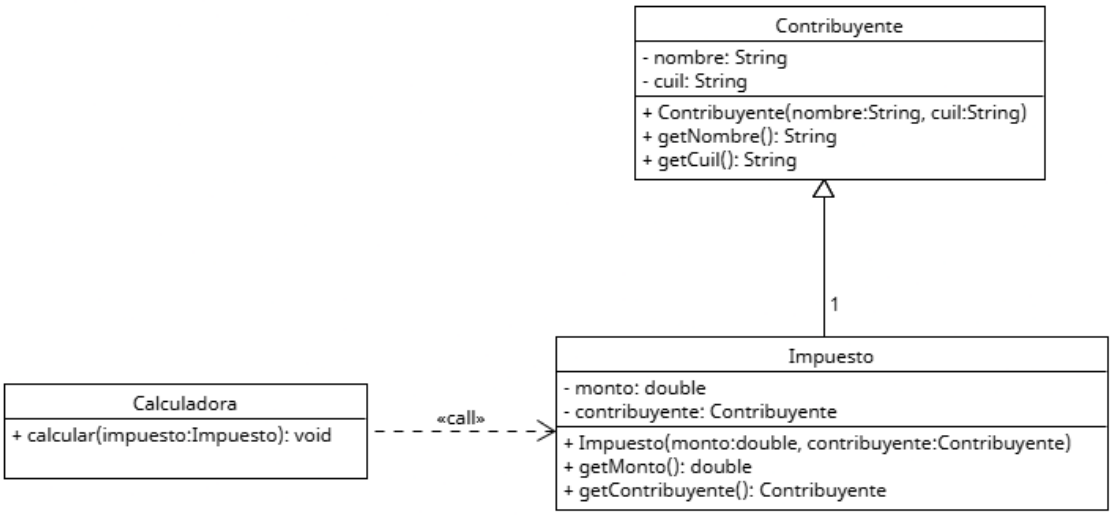
```
public class TP5 {
    public static void main(String[] args) {
        Artista artista = new Artista("Charly Garcia", "Rock");
        Cancion cancion = new Cancion("Rezo por vos", artista);

        Reproductor reproductor = new Reproductor();
        reproductor.reproducir(cancion);
    }
}
```

12. Impuesto - Contribuyente - Calculadora

a. Asociación unidireccional: Impuesto → Contribuyente

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

Diagrama



Código

Clase 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 String getCuil() {
        return cuil;
```

```
    }

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

Clase Impuesto

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

    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=" + contribuyente + '}';
    }
}
```

Clase Calculadora

```
public class Calculadora {
    public void calcular(Impuesto impuesto) {
        double total = impuesto.getMonto() * 1.21;
        System.out.println("Calculando impuesto para: " + impuesto.getContribuyente().getNombre());
        System.out.println("Monto original: " + impuesto.getMonto());
        System.out.println("Monto con IVA: " + total);
    }
}
```

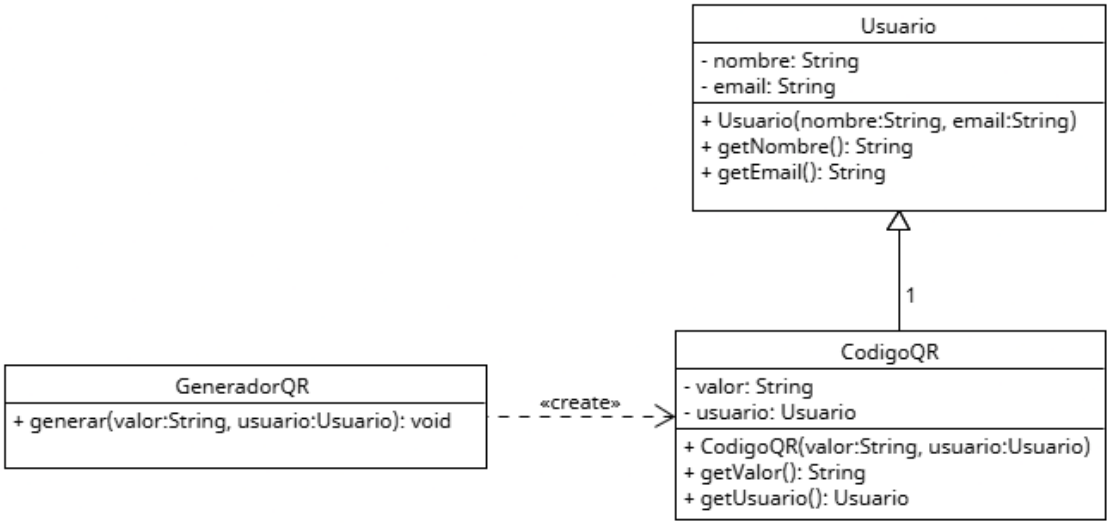
Main

```
public class TP9 {
    public static void main(String[] args) {
        Contribuyente contribuyente = new Contribuyente("María López", "20-12345678-9");
        Impuesto impuesto = new Impuesto(10000.0, contribuyente);

        Calculadora calculadora = new Calculadora();
        calculadora.calcular(impuesto);
    }
}
```

13. GeneradorQR - Usuario - CódigoQR

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

Diagrama



Código

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;
    }

    @Override
```

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

CodigoQR

```
public class CodigoQR {
    private String valor;
    private Usuario usuario;

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

    public String getValor() {
        return valor;
    }

    public Usuario getUsuario() {
        return usuario;
    }

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

GeneradorQR

```
public class GeneradorQR {
    public void generar(String valor, Usuario usuario) {
        CodigoQR qr = new CodigoQR(valor, usuario);
```

```
        System.out.println("QR generado: " + qr);
    }
}
```

Main

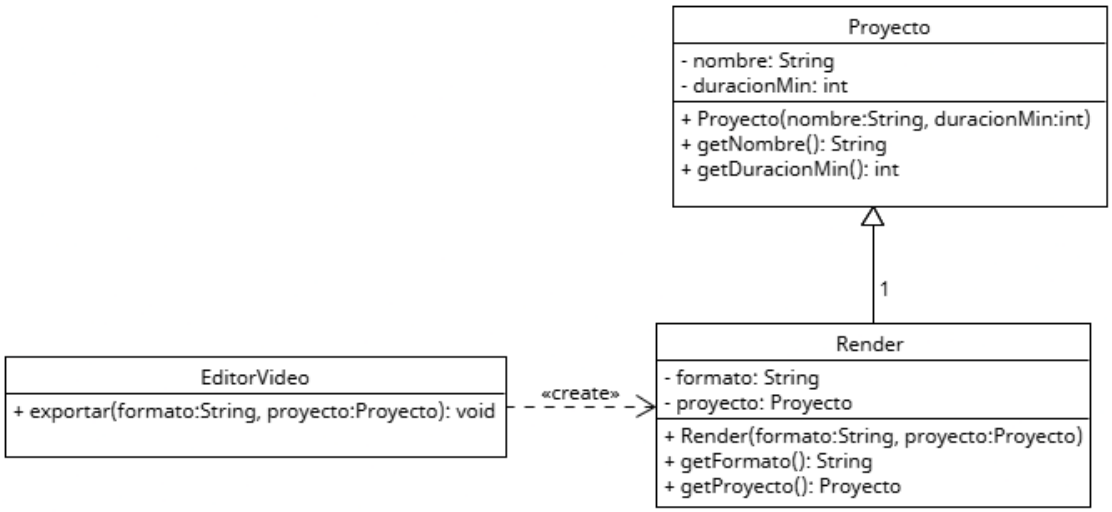
```
public class TP5 {
    public static void main(String[] args) {
        Usuario usuario = new Usuario("Juan Peres", "juanperez@hotmail.com");
        GeneradorQR generador = new GeneradorQR();

        generador.generar("SXC5D55130SD", usuario);
    }
}
```

14. EditorVideo - Proyecto - Render

a. Asociación unidireccional: Render → Proyecto

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

Diagrama



Código

Proyecto

```
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 + ", duracionMin=" + duracionMin + '}';
    }
}
```

Render

```
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=" + proyecto + '}';
    }
}
```

Editor video

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

Main

```
public class TP14 {
    public static void main(String[] args) {
        Proyecto proyecto = new Proyecto("Produccion final", 12);
        EditorVideo editor = new EditorVideo();

        editor.exportar("MP4", proyecto);
    }
}
```