Programación II - TP5: Relaciones UML 1 a 1

Alumno: *Romero, Abel Tomás (Comisión 5)*
Link del repo de GitHub:
https://github.com/Tomu98/UTN-TUPaD-P2-TPs/tree/main/05%20UML%20B%C3%A1sico

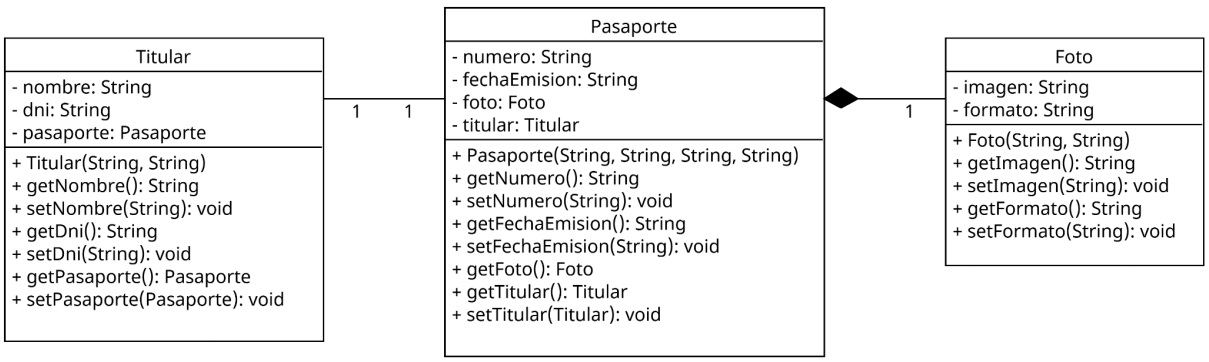
Objetivo General:
Modelar clases con relaciones 1 a 1 utilizando diagramas UML. Identificar correctamente el tipo de relación (asociación, agregación, composición, dependencia) y su dirección, y llevarlas a implementación en Java.

Caso Práctico:
Desarrollar los siguientes ejercicios en Java. Cada uno deberá incluir:

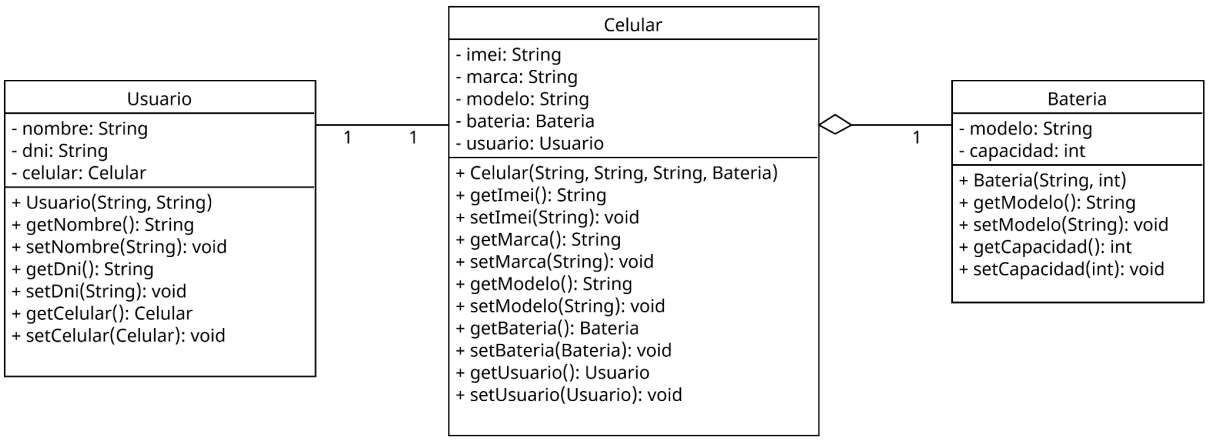
- Diagrama UML.
- Tipo de relación (asociación, agregación, composición, dependencia).
- Dirección (unidireccional o bidireccional).
- Implementación de las clases con atributos y relaciones definidas.

Ejercicios de Relaciones 1 a 1:

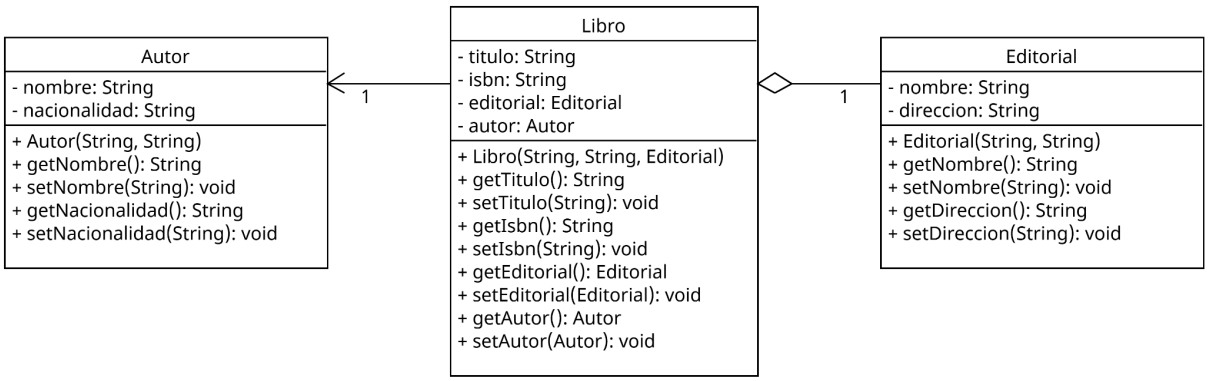
1. **Pasaporte - Foto - Titular**
    a. Composición: **Pasaporte → Foto**
    b. Asociación bidireccional: **Pasaporte ↔ Titular**
    c. **Clases y atributos:**
        - Pasaporte: numero, fechaEmision
        - Foto: imagen, formato
        - Titular: nombre, dni



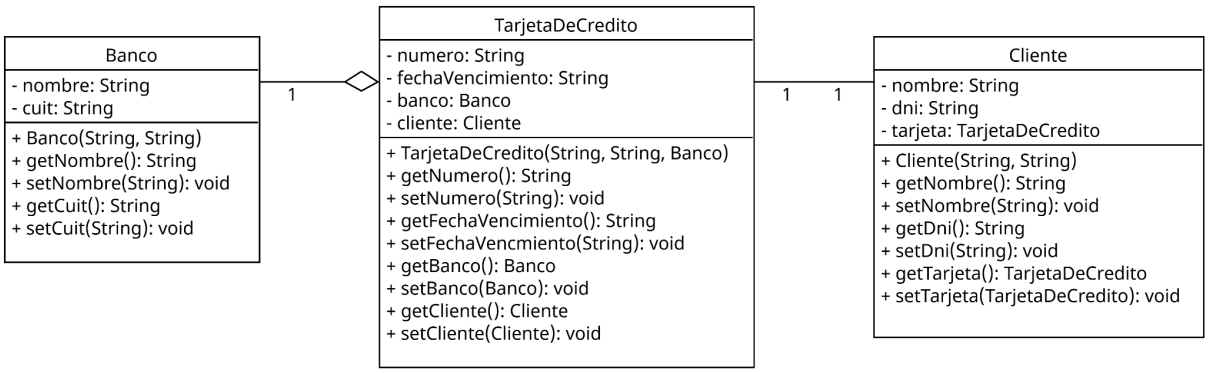
2. **Celular - Bateria - Usuario**
    a. Agregación: **Celular → Bateria**
    b. Asociación bidireccional: **Celular ↔ Usuario**
    c. **Clases y atributos:**
        - Celular: imei, marca, modelo
        - Bateria: modelo, capacidad
        - Usuario: nombre, dni



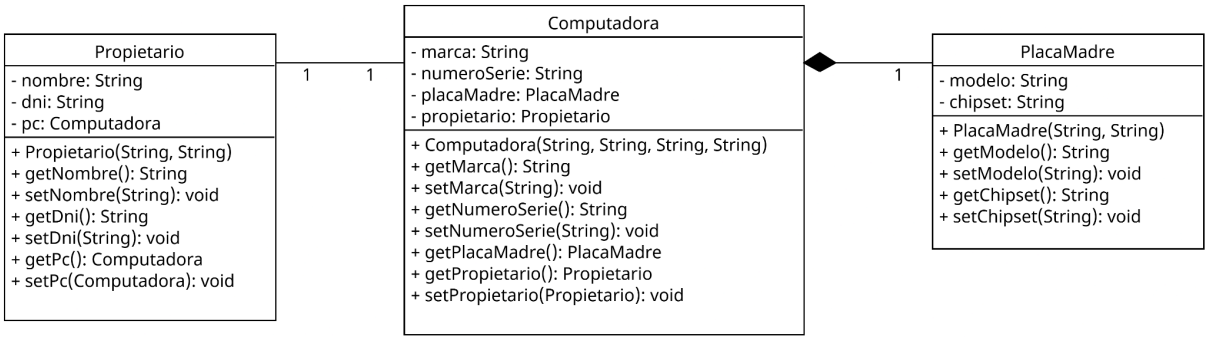
3. **Libro - Autor - Editorial**
    a. Agregación: **Libro → Editorial**
    b. Asociación unidireccional: **Libro → Autor**
    c. **Clases y atributos:**
        - Libro: titulo, isbn
        - Autor: nombre, nacionalidad
        - Editorial: nombre, direccion



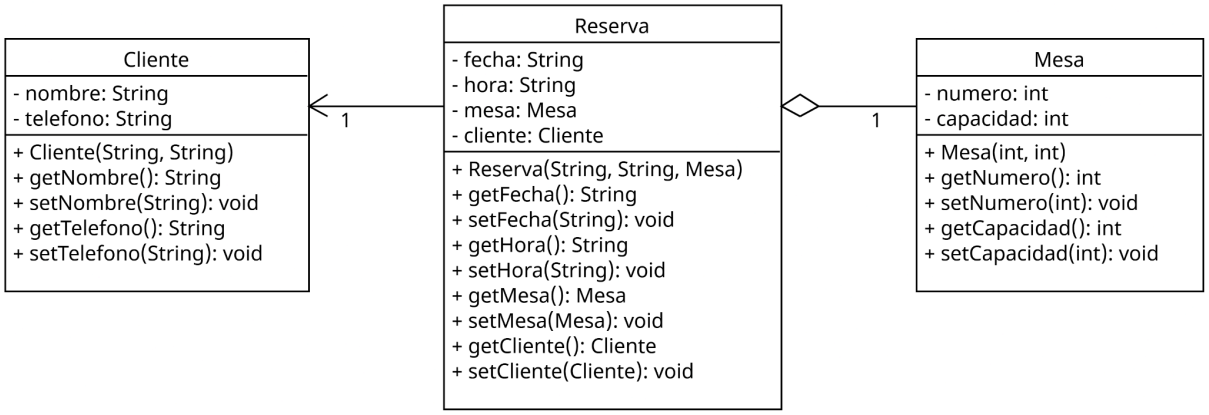
4. **TarjetaDeCrédito - Cliente - Banco**
    a. Agregación: **TarjetaDeCredito → Banco**
    b. Asociación bidireccional: **TarjetaDeCredito ↔ Cliente**
    c. **Clases y atributos:**
        - TarjetaDeCredito: numero, fechaVencimiento
        - Cliente: nombre, dni
        - Banco: nombre, cuit



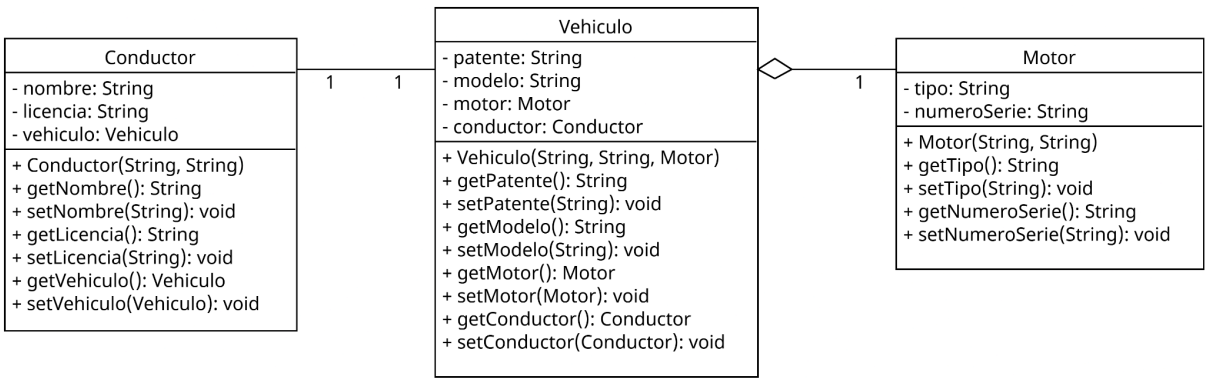
5. **Computadora - PlacaMadre - Propietario**
    a. Composición: **Computadora → PlacaMadre**
    b. Asociación bidireccional: **Computadora ↔ Propietario**
    c. **Clases y atributos:**
        - Computadora: marca, numeroSerie
        - PlacaMadre: modelo, chipset
        - Propietario: nombre, dni



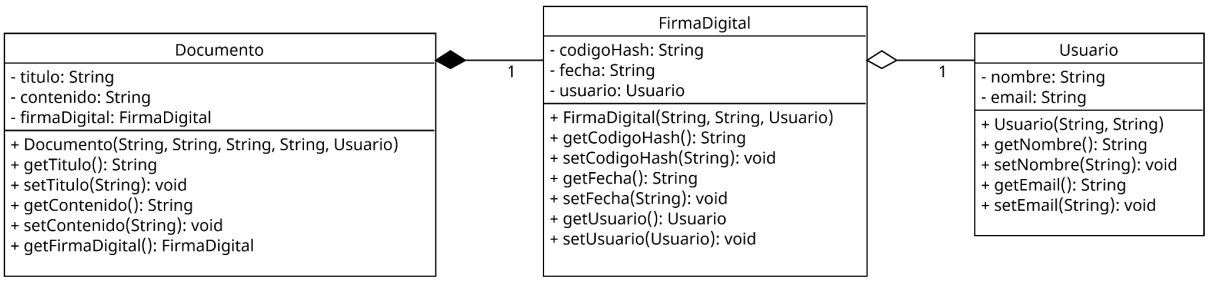
6. **Reserva - Cliente - Mesa**
    a. Agregación: **Reserva → Mesa**
    b. Asociación unidireccional: **Reserva → Cliente**
    c. **Clases y atributos:**
        - Reserva: fecha, hora
        - Cliente: nombre, telefono
        - Mesa: numero, capacidad



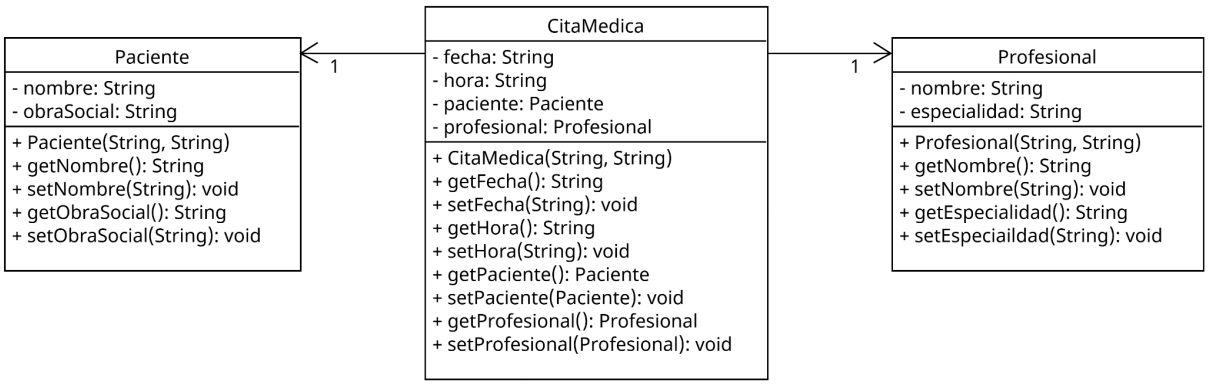
7. **Vehiculo - Motor - Conductor**
    a. Agregación: **Vehiculo → Motor**
    b. Asociación bidireccional: **Vehiculo ↔ Conductor**
    c. **Clases y atributos:**
        - Vehiculo: patente, modelo
        - Motor: tipo, numeroSerie
        - Conductor: nombre, licencia



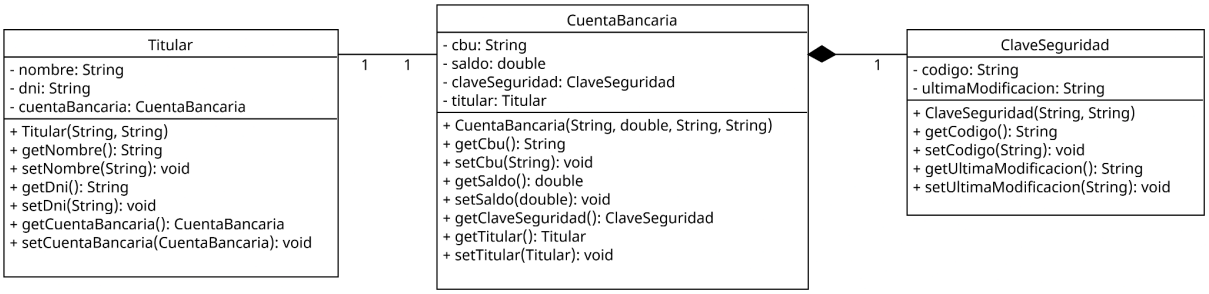
8. **Documento - FirmaDigital - Usuario**
    a. Composición: **Documento → FirmaDigital**
    b. Agregación: **FirmaDigital → Usuario**
    c. **Clases y atributos:**
        - Documento: titulo, contenido
        - FirmaDigital: codigoHash, fecha
        - Usuario: nombre, email



9. **CitaMedica - Paciente - Profesional**
    a. Asociación unidireccional: **CitaMedica → Paciente**
    b. Asociación unidireccional: **CitaMedica → Profesional**
    c. **Clases y atributos:**
        - CitaMedica: fecha, hora
        - Paciente: nombre, obraSocial
        - Profesional: nombre, especialidad



10. **CuentaBancaria - ClaveSeguridad - Titular**
    a. Composición: **CuentaBancaria → ClaveSeguridad**
    b. Asociación bidireccional: **CuentaBancaria ↔ Titular**
    c. **Clases y atributos:**
        - CuentaBancaria: cbu, saldo
        - ClaveSeguridad: codigo, ultimaModificacion
        - Titular: nombre, dni

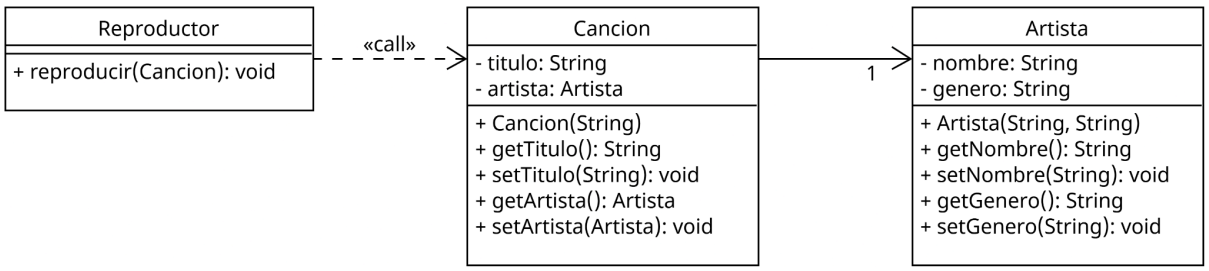


DEPENDENCIA DE USO

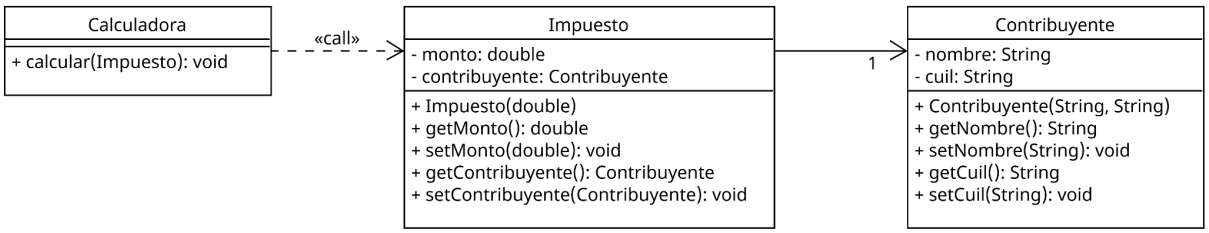
La clase usa otra como parámetro de un método, pero no la guarda como atributo.

Ejercicios de Dependencia de Uso

11. **Reproductor - Cancion - Artista**
    a. Asociación unidireccional: **Cancion → Artista**
    b. Dependencia de uso: **Reproductor.reproducir(Cancion)**
    c. **Clases y atributos:**
        - Cancion: titulo
        - Artista: nombre, genero
        - Reproductor->método: void reproducir(Cancion cancion)



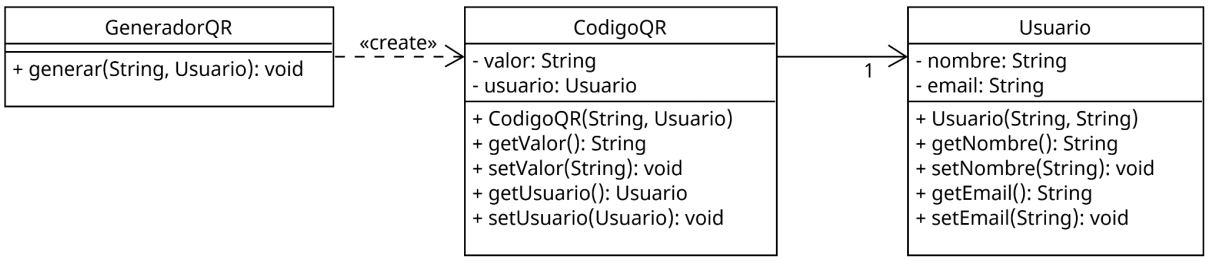
12. **Impuesto - Contribuyente - Calculadora**
    a. Asociación unidireccional: **Impuesto → Contribuyente**
    b. Dependencia de uso: **Calculadora.calcular(Impuesto)**
    c. **Clases y atributos:**
        - Impuesto: monto
        - Contribuyente: nombre, cuil
        - Calculadora->método: void 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 - CodigoQR**
    a. Asociación unidireccional: **CodigoQR → Usuario**
    b. Dependencia de creación: **GeneradorQR.generar(String, Usuario)**
    c. **Clases y atributos:**
        - CodigoQR: valor
        - Usuario: nombre, email
        - 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:**
        - Render: formato
        - Proyecto: nombre, duracionMin
        - EditorVideo->método: void exportar(String formato, Proyecto proyecto)