Trabajo Practico N°5: UML

Tiziano Nahuel Caamaño

Ejercicios de Relaciones 1 a 1

1. Pasaporte - Foto – Titular

UML:

Java:

Foto.java

```java
public class Foto {  no usages
    private String imagen; // Simulado con un String, podría ser un array de bytes  2 usages
    private String formato;  2 usages

    public Foto(String imagen, String formato) {  no usages
        this.imagen = imagen;
        this.formato = formato;
    }

    @Override
    public String toString() {
        return "Foto [formato=" + formato + ", imagen=" + imagen + "]";
    }
}
```

Titular.java

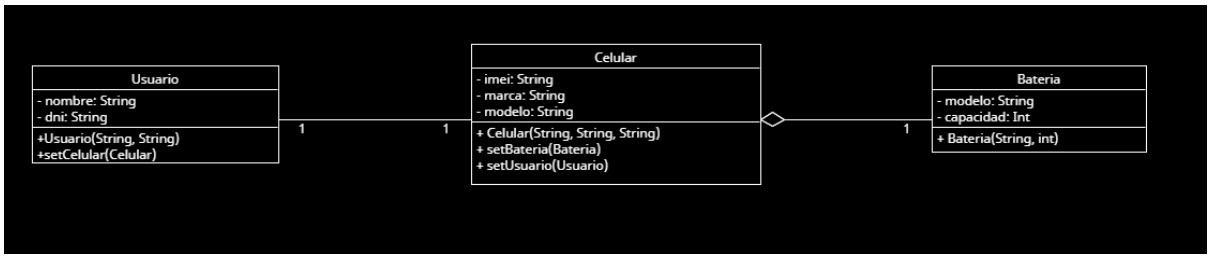
```java
public class Titular {  no usages
    private String nombre;  3 usages
    private String dni;  2 usages
    private Pasaporte pasaporte; // Referencia al Pasaporte  3 usages

    public Titular(String nombre, String dni) {  no usages
        this.nombre = nombre;
        this.dni = dni;
    }

    // Método para establecer la relación bidireccional
    public void setPasaporte(Pasaporte pasaporte) {  no usages
        this.pasaporte = pasaporte;
    }

    @Override
    public String toString() {
        return "Titular [nombre=" + nombre + ", dni=" + dni + ", numeroPasaporte=" + (pasaporte != null ? pasaporte.getNumero() : "N/A") + "]";
    }

    public String getNombre() {  no usages
        return nombre;
    }
}
```

Pasaporte.java

```java
import java.time.LocalDate;

public class Pasaporte {  2 usages
    private String numero;  3 usages
    private LocalDate fechaEmision;  2 usages
    private Foto foto; // Composición: la foto es parte del pasaporte  2 usages
    private Titular titular; // Asociación: referencia al titular  3 usages

    // El objeto Foto se crea junto con el Pasaporte, demostrando la composición
    public Pasaporte(String numero, LocalDate fechaEmision, String urlImagen, String formatoFoto) {  no usages
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = new Foto(urlImagen, formatoFoto); // El ciclo de vida de Foto depende de Pasaporte
    }

    // Método para establecer la relación bidireccional
    public void setTitular(Titular titular) {  no usages
        this.titular = titular;
        titular.setPasaporte(this); // Se establece la referencia en la otra dirección
    }

    public String getNumero() {  1 usage
        return numero;
    }

    @Override
    public String toString() {
        return "Pasaporte [numero=" + numero + ", fechaEmision=" + fechaEmision + ", foto=" + foto + ", titular=" + (titular != null ? titular.getNombre() : "N/A") + "]";
    }
}
```

2. Celular - Batería - Usuario

UML:



Java:

Celular.java

```java
public class Celular {  no usages
    private String imei;  1 usage
    private String marca;  2 usages
    private String modelo;  3 usages
    private Bateria bateria; // Agregación  2 usages
    private Usuario usuario; // Asociación  3 usages

    public Celular(String imei, String marca, String modelo) {  no usages
        this.imei = imei;
        this.marca = marca;
        this.modelo = modelo;
    }

    // El objeto Bateria se recibe desde afuera, demostrando agregación
    public void setBateria(Bateria bateria) {  no usages
        this.bateria = bateria;
    }

    // Método para la relación bidireccional
    public void setUsuario(Usuario usuario) {  no usages
        this.usuario = usuario;
        usuario.setCelular(this);
    }

    public String getModelo() {  no usages
        return modelo;
    }

    @Override
    public String toString() {
        return "Celular [marca=" + marca + ", modelo=" + modelo + ", bateria=" + bateria + ", usuario=" + (usuario != null ? usuario.getNombre() : "N/A") + "]";
    }
}
```

Bateria.java

```java
public class Bateria {  2 usages
    private String modelo;  2 usages
    private int capacidad; // en mAh  2 usages

    public Bateria(String modelo, int capacidad) {  no usages
        this.modelo = modelo;
        this.capacidad = capacidad;
    }

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

Usuario.java

```java
public class Usuario {  2 usages
    private String nombre;  3 usages
    private String dni;  2 usages
    private Celular celular;  3 usages

    public Usuario(String nombre, String dni) {  no usages
        this.nombre = nombre;
        this.dni = dni;
    }

    public void setCelular(Celular celular) {  1 usage
        this.celular = celular;
    }

    public String getNombre() {  1 usage
        return nombre;
    }

    @Override
    public String toString() {
        return "Usuario [nombre=" + nombre + ", dni=" + dni + ", celular=" + (celular != null ? celular.getModelo() : "N/A") + "]";
    }
}
```

3. Libro - Autor – Editorial

UML:

Java:

Libro.java

```java
public class Libro {  no usages
    private String titulo;  2 usages
    private String isbn;  2 usages
    private Autor autor; // Asociación unidireccional  2 usages
    private Editorial editorial; // Agregación  2 usages

    // Los objetos Autor y Editorial se crean fuera y se pasan al Libro
    public Libro(String titulo, String isbn, Autor autor, Editorial editorial) {  no usages
        this.titulo = titulo;
        this.isbn = isbn;
        this.autor = autor;
        this.editorial = editorial;
    }

    @Override
    public String toString() {
        return "Libro [titulo=" + titulo + ", isbn=" + isbn + "]\n --> " + autor + "\n --> " + editorial;
    }
}
```

Editorial.java

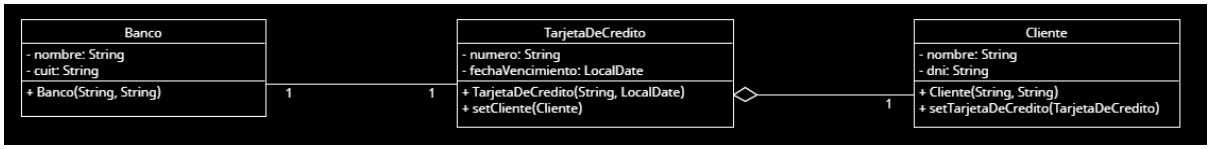
```java
public class Editorial {
    private String nombre;  2 usages
    private String direccion;  2 usages

    public Editorial(String nombre, String direccion) {  no usages
        this.nombre = nombre;
        this.direccion = direccion;
    }

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

Autor.java

```java
public class Autor {  2 usages
    private String nombre;  2 usages
    private String nacionalidad;  2 usages

    public Autor(String nombre, String nacionalidad) {  no usages
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }

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

4. TarjetaDeCrédito - Cliente – Banco

UML:



Java:

TarjetaDeCredito.java

```java
import java.time.LocalDate;

public class TarjetaDeCredito {  no usages
    private String numero;  3 usages
    private LocalDate fechaVencimiento;  2 usages
    private Cliente cliente; // Asociación  3 usages
    private Banco banco;     // Agregación  2 usages

    public TarjetaDeCredito(String numero, LocalDate fechaVencimiento, Banco banco) {  no usages
        this.numero = numero;
        this.fechaVencimiento = fechaVencimiento;
        this.banco = banco;
    }

    public void setCliente(Cliente cliente) {  no usages
        this.cliente = cliente;
        if (cliente != null) {
            cliente.setTarjetaDeCredito(this); // Establece la relación inversa
        }
    }

    public String getNumero() {  no usages
        return numero;
    }

    @Override
    public String toString() {
        return "TarjetaDeCredito [numero=" + numero + ", fechaVencimiento=" + fechaVencimiento +
                ", cliente=" + (cliente != null ? cliente.getNombre() : "N/A") + ", banco=" + banco + "]";
    }
}
```

Cliente.java

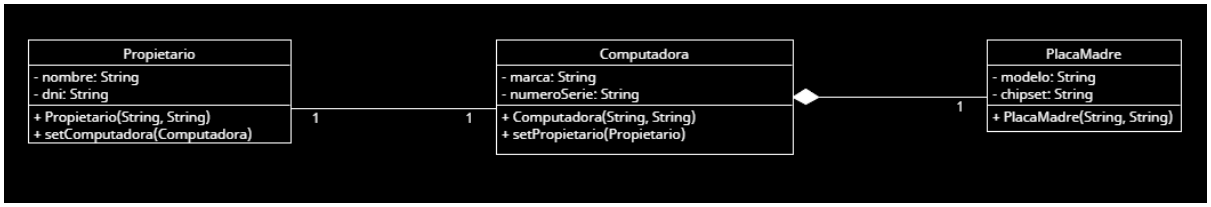
```java
public class Cliente {  2 usages
    private String nombre;  3 usages
    private String dni;  2 usages
    private TarjetaDeCredito tarjeta;  3 usages

    public Cliente(String nombre, String dni) {  no usages
        this.nombre = nombre;
        this.dni = dni;
    }

    public void setTarjetaDeCredito(TarjetaDeCredito tarjeta) {  1 usage
        this.tarjeta = tarjeta;
    }

    public String getNombre() {  1 usage
        return nombre;
    }

    @Override
    public String toString() {
        return "Cliente [nombre=" + nombre + ", dni=" + dni + ", tarjeta=" + (tarjeta != null ? tarjeta.getNumero() : "N/A") + "]";
    }
}
```

Banco.java

```java
public class Banco {  2 usages
    private String nombre;  2 usages
    private String cuit;  2 usages

    public Banco(String nombre, String cuit) {  no usages
        this.nombre = nombre;
        this.cuit = cuit;
    }

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

5. Computadora - PlacaMadre – Propietario

UML:



Java:

Computadora.java

```java
public class Computadora {  2 usages
    private String marca;  2 usages
    private String numeroSerie;  2 usages
    private PlacaMadre placaMadre; // Composición  2 usages
    private Propietario propietario; // Asociación  3 usages

    public Computadora(String marca, String numeroSerie, String modeloPlaca, String chipsetPlaca) {  no usages
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        // La PlacaMadre se crea aquí, su vida depende de la Computadora
        this.placaMadre = new PlacaMadre(modeloPlaca, chipsetPlaca);
    }

    public void setPropietario(Propietario propietario) {  no usages
        this.propietario = propietario;
        if (propietario != null) {
            propietario.setComputadora(this);
        }
    }

    @Override
    public String toString() {
        return "Computadora [marca=" + marca + ", numeroSerie=" + numeroSerie +
                ", placaMadre=" + placaMadre + ", propietario=" + (propietario != null ? propietario.getNombre() : "N/A") + "]";
    }

}
```

PlacaMadre.java

```java
public class PlacaMadre {  2 usages
    private String modelo;  2 usages
    private String chipset;  2 usages

    public PlacaMadre(String modelo, String chipset) {  1 usage
        this.modelo = modelo;
        this.chipset = chipset;
    }

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

Propietario.java

```java
public class Propietario {  2 usages
    private String nombre;  3 usages
    private String dni;  2 usages
    private Computadora computadora;  1 usage

    public Propietario(String nombre, String dni) {  no usages
        this.nombre = nombre;
        this.dni = dni;
    }

    public void setComputadora(Computadora computadora) {  1 usage
        this.computadora = computadora;
    }

    public String getNombre() {  1 usage
        return nombre;
    }

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

6. Reserva - Cliente – Mesa

UML:

Java:

Reserva.java

```java
import java.time.LocalDate;
import java.time.LocalTime;

public class Reserva {  no usages
    private LocalDate fecha;  2 usages
    private LocalTime hora;  2 usages
    private Cliente cliente; // Asociación Unidireccional  2 usages
    private Mesa mesa;       // Agregación Unidireccional  2 usages

    public Reserva(LocalDate fecha, LocalTime hora, Cliente cliente, Mesa mesa) {  no usages
        this.fecha = fecha;
        this.hora = hora;
        this.cliente = cliente;
        this.mesa = mesa;
    }

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

Cliente.java

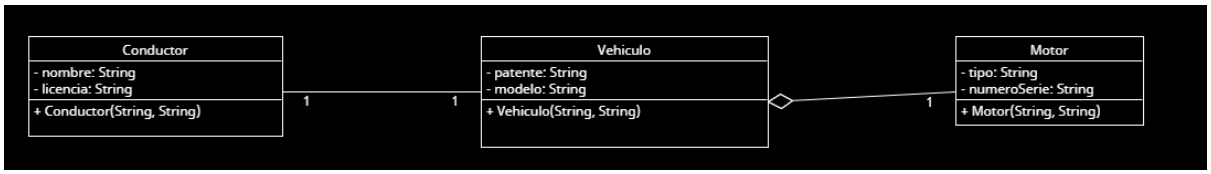
```java
public class Cliente {  no usages
    private String nombre;  2 usages
    private String telefono;  2 usages

    public Cliente(String nombre, String telefono) {  no usages
        this.nombre = nombre;
        this.telefono = telefono;
    }

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

Mesa.java

```java
public class Mesa {  no usages
    private int numero;  2 usages
    private int capacidad;  2 usages

    public Mesa(int numero, int capacidad) {  no usages
        this.numero = numero;
        this.capacidad = capacidad;
    }

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

7. Vehículo - Motor – Conductor

UML:



Java:

Vehiculo.java

```java
public class Vehiculo {  2 usages
    private String patente;  2 usages
    private String modelo;  2 usages
    private Motor motor; // Agregación  2 usages
    private Conductor conductor; // Asociación  3 usages

    public Vehiculo(String patente, String modelo) {  no usages
        this.patente = patente;
        this.modelo = modelo;
    }

    public void setMotor(Motor motor) {  no usages
        this.motor = motor;
    }

    public void setConductor(Conductor conductor) {  no usages
        this.conductor = conductor;
        if (conductor != null) {
            conductor.setVehiculo(this);
        }
    }

    @Override
    public String toString() {
        return "Vehiculo [patente=" + patente + ", modelo=" + modelo + ", motor=" + motor +
                ", conductor=" + (conductor != null ? conductor.getNombre() : "N/A") + "]";
    }
}
```

Motor.java

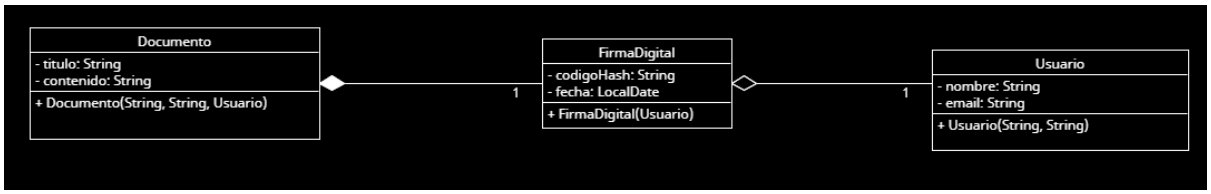
```java
public class Motor {  no usages
    private String tipo;  2 usages
    private String numeroSerie;  2 usages

    public Motor(String tipo, String numeroSerie) {  no usages
        this.tipo = tipo;
        this.numeroSerie = numeroSerie;
    }

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

Conductor.java

```java
public class Conductor {  no usages
    private String nombre;  3 usages
    private String licencia;  2 usages
    private Vehiculo vehiculo;  1 usage

    public Conductor(String nombre, String licencia) {  no usages
        this.nombre = nombre;
        this.licencia = licencia;
    }

    public void setVehiculo(Vehiculo vehiculo) {  no usages
        this.vehiculo = vehiculo;
    }

    public String getNombre() {  no usages
        return nombre;
    }

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

8. Documento - FirmaDigital – Usuario

UML:



Java:

Documento.java

```java
public class Documento {  no usages
    private String titulo;  2 usages
    private String contenido;  1 usage
    private FirmaDigital firmaDigital; // Composición  2 usages

    public Documento(String titulo, String contenido, Usuario firmante) {  no usages
        this.titulo = titulo;
        this.contenido = contenido;
        // La FirmaDigital es creada y gestionada por el Documento.
        this.firmaDigital = new FirmaDigital(firmante);
    }

    @Override
    public String toString() {
        return "Documento [titulo=" + titulo + "]\n --> " + firmaDigital;
    }
}
```

FirmaDigital.java

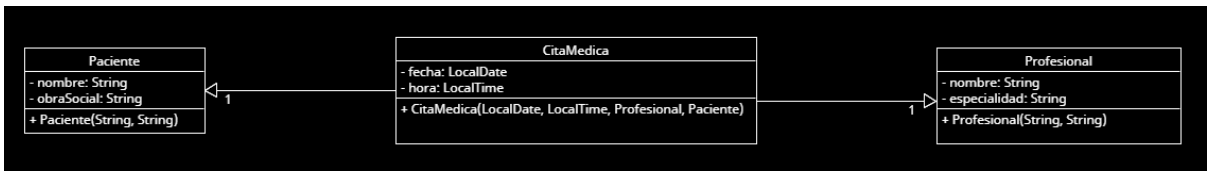
```java
import java.time.LocalDate;
import java.util.UUID;

public class FirmaDigital {  no usages
    private String codigoHash;  2 usages
    private LocalDate fecha;  2 usages
    private Usuario usuario; // Agregación  2 usages

    // La firma se crea para un usuario específico.
    public FirmaDigital(Usuario usuario) {  no usages
        this.usuario = usuario;
        this.fecha = LocalDate.now();
        // Se genera un hash único para simular la firma.
        this.codigoHash = UUID.randomUUID().toString();
    }

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

Usuario.java

```java
public class Usuario {  no usages
    private String nombre;  2 usages
    private String email;  2 usages

    public Usuario(String nombre, String email) {  no usages
        this.nombre = nombre;
        this.email = email;
    }

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

9. CitaMédica - Paciente – Profesional

UML:



Java:

CitaMedica.java

```java
import java.time.LocalDate;
import java.time.LocalTime;

public class CitaMedica {  no usages
    private LocalDate fecha;  2 usages
    private LocalTime hora;  2 usages
    private Paciente paciente;  2 usages
    private Profesional profesional;  2 usages

    public CitaMedica(LocalDate fecha, LocalTime hora, Paciente paciente, Profesional profesional) {  no usages
        this.fecha = fecha;
        this.hora = hora;
        this.paciente = paciente;
        this.profesional = profesional;
    }

    @Override
    public String toString() {
        return "CitaMedica [fecha=" + fecha + ", hora=" + hora + "]\n --> " + paciente + "\n --> " + profesional;
    }
}
```

Paciente.java

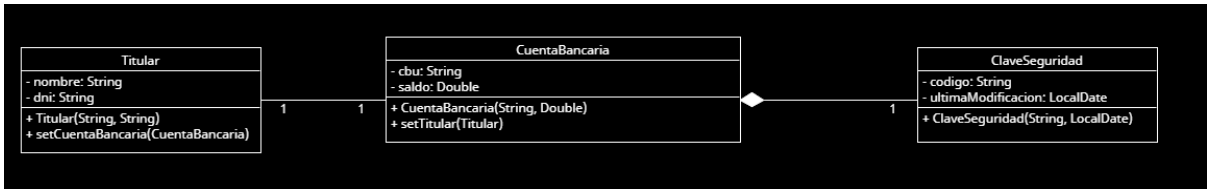
```java
public class Paciente {  2 usages
    private String nombre;  2 usages
    private String obraSocial;  2 usages

    public Paciente(String nombre, String obraSocial) {  no usages
        this.nombre = nombre;
        this.obraSocial = obraSocial;
    }

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

Profesional.java

```java
public class Profesional {  2 usages
    private String nombre;  2 usages
    private String especialidad;  2 usages

    public Profesional(String nombre, String especialidad) {  no usages
        this.nombre = nombre;
        this.especialidad = especialidad;
    }

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

10. CuentaBancaria - ClaveSeguridad – Titular

UML:



Java:

CuentaBancaria.java

```java
public class CuentaBancaria {  2 usages
    private String cbu;  2 usages
    private double saldo;  2 usages
    private ClaveSeguridad clave; // Composición  2 usages
    private Titular titular;      // Asociación  3 usages

    public CuentaBancaria(String cbu, double saldo, String codigoClave) {  no usages
        this.cbu = cbu;
        this.saldo = saldo;
        // La ClaveSeguridad se crea y vive dentro de la CuentaBancaria
        this.clave = new ClaveSeguridad(codigoClave);
    }

    public void setTitular(Titular titular) {  no usages
        this.titular = titular;
        if (titular != null) {
            titular.setCuentaBancaria(this);
        }
    }

    @Override
    public String toString() {
        return "CuentaBancaria [cbu=" + cbu + ", saldo=" + saldo + ", clave=" + clave +
                ", titular=" + (titular != null ? titular.getNombre() : "N/A") + "]";
    }
}
```

ClaveSeguridad.java

```java
import java.time.LocalDate;

public class ClaveSeguridad {  no usages
    private String codigo;  1 usage
    private LocalDate ultimaModificacion;  2 usages

    public ClaveSeguridad(String codigo) {  no usages
        this.codigo = codigo;
        this.ultimaModificacion = LocalDate.now();
    }

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

Titular.java

```java
public class Titular {  no usages
    private String nombre;  3 usages
    private String dni;  2 usages
    private CuentaBancaria cuenta;  1 usage

    public Titular(String nombre, String dni) {  no usages
        this.nombre = nombre;
        this.dni = dni;
    }

    public void setCuentaBancaria(CuentaBancaria cuenta) {  no usages
        this.cuenta = cuenta;
    }

    public String getNombre() {  no usages
        return nombre;
    }

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

UML:

Java:

Reproductor.Java

```java
public class Reproductor {  no usages
    // Dependencia de Uso: 'cancion' solo existe dentro de este método.
    public void reproducir(Cancion cancion) {  no usages
        System.out.println("Reproduciendo ahora: " + cancion.toString());
    }
}
```

Cancion.Java

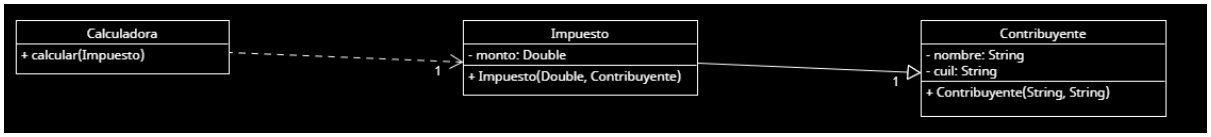
```java
public class Cancion {  no usages
    private String titulo;  2 usages
    private Artista artista; // Asociación  2 usages

    public Cancion(String titulo, Artista artista) {  no usages
        this.titulo = titulo;
        this.artista = artista;
    }

    @Override
    public String toString() {
        return "'" + titulo + "' de " + artista.toString();
    }
}
```

Artista.Java

```java
public class Artista {  no usages
    private String nombre;  2 usages
    private String genero;  2 usages

    public Artista(String nombre, String genero) {  no usages
        this.nombre = nombre;
        this.genero = genero;
    }

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

12. Impuesto - Contribuyente – Calculadora

UML:



Java:

Impuesto.Java

```java
public class Impuesto {  no usages
    private double monto;  3 usages
    private Contribuyente contribuyente; // Asociación  2 usages

    public Impuesto(double monto, Contribuyente contribuyente) {  no usages
        this.monto = monto;
        this.contribuyente = contribuyente;
    }

    public double getMonto() {  no usages
        return monto;
    }

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

Contribuyente.Java

```java
public class Contribuyente {  no usages
    private String nombre;  2 usages
    private String cuil;  2 usages

    public Contribuyente(String nombre, String cuil) {  no usages
        this.nombre = nombre;
        this.cuil = cuil;
    }

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

Calculadora.Java

```java
public class Calculadora {  no usages
    // Dependencia de Uso: 'impuesto' es un parámetro temporal.
    public void calcular(Impuesto impuesto) {  no usages
        // Simula un cálculo complejo.
        double total = impuesto.getMonto() * 1.21;
        System.out.println("Calculando impuesto para: " + impuesto.toString());
        System.out.println("Total a pagar: $" + total);
    }
}
```

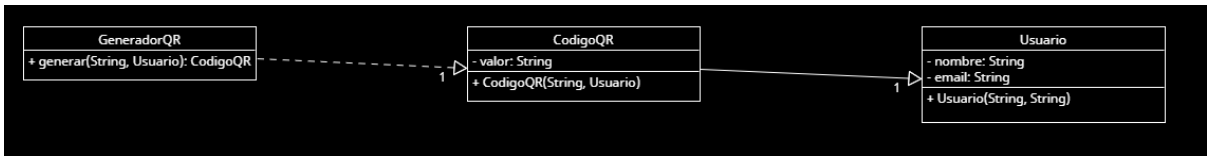
DEPENDENCIA DE CREACIÓN

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

Ejercicios de Dependencia de Creación

13. GeneradorQR - Usuario - CódigoQR

UML:



Java:

GeneradorQR.Java

```java
public class GeneradorQR {  no usages
    // Dependencia de Creación: El método 'generar' crea y devuelve un CodigoQR.
    public CodigoQR generar(String valor, Usuario usuario) {  no usages
        // Crea la instancia localmente.
        CodigoQR nuevoCodigo = new CodigoQR(valor, usuario);
        System.out.println("Generando nuevo código QR...");
        return nuevoCodigo;
    }
}
```

Usuario.Java

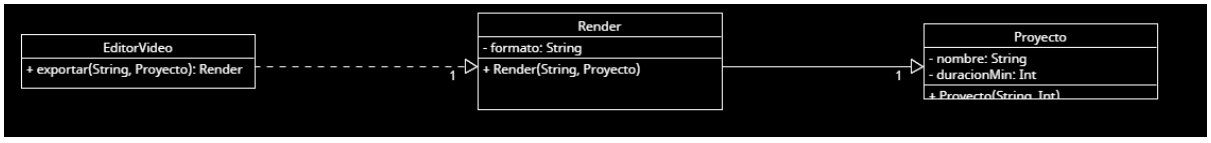
```java
public class Usuario {  no usages
    private String nombre;  2 usages
    private String email;  2 usages

    public Usuario(String nombre, String email) {  no usages
        this.nombre = nombre;
        this.email = email;
    }

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

CodigoQR.Java

```java
public class CodigoQR {  no usages
    private String valor;  2 usages
    private Usuario usuario; // Asociación  2 usages

    public CodigoQR(String valor, Usuario usuario) {  no usages
        this.valor = valor;
        this.usuario = usuario;
    }

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

14. EditorVideo - Proyecto – Render

UML:



Java:

EditorVideo.Java

```java
public class EditorVideo {  no usages
    // Dependencia de Creación: El método 'exportar' crea una instancia de Render.
    public Render exportar(String formato, Proyecto proyecto) {  no usages
        System.out.println("Iniciando exportación del proyecto: " + proyecto.toString());
        // El objeto Render es creado aquí.
        Render renderFinal = new Render(formato, proyecto);
        System.out.println("Render finalizado con éxito.");
        return renderFinal;
    }
}
```

Proyecto.Java

```java
public class Proyecto {  no usages
    private String nombre;  2 usages
    private int duracionMin;  2 usages

    public Proyecto(String nombre, int duracionMin) {  no usages
        this.nombre = nombre;
        this.duracionMin = duracionMin;
    }

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

Render.Java

```java
public class Render {  no usages
    private String formato;  2 usages
    private Proyecto proyecto; // Asociación  2 usages

    public Render(String formato, Proyecto proyecto) {  no usages
        this.formato = formato;
        this.proyecto = proyecto;
    }

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