TRABAJO PRACTICO Nº5 UML

Nombre: Agustina

Apellido: Cruz

Comisión: Nº12.

Materia: Programación II

Profesor: Ariel Enferrel

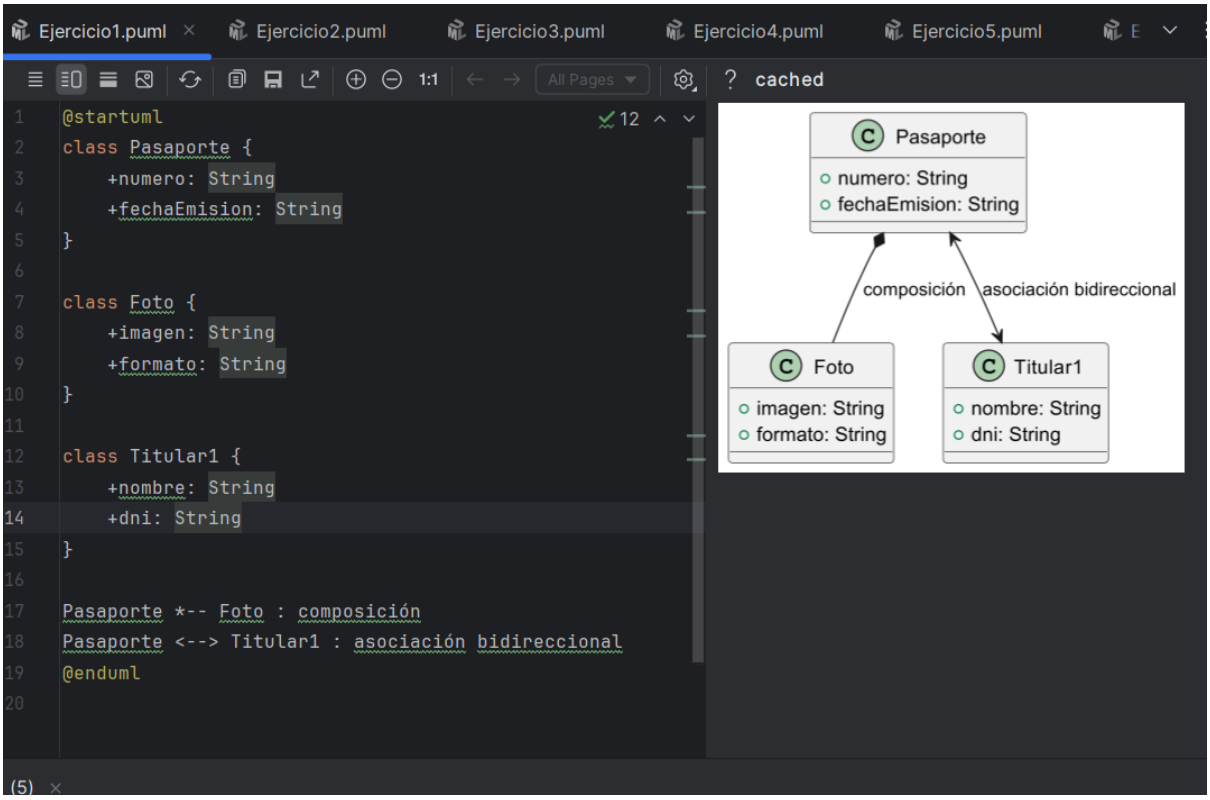
Año: 2025

Índice

Ejercicios

Ejercicio 1 – Pasaporte, Foto, Titular

Diagrama UML con Su código:



Archivos: Pasaporte.java, Foto.java, Titular.java
Relaciones:

- Composición: Pasaporte → Foto
- Asociación bidireccional: Pasaporte ↔ Titular

Código Java: public class Pasaporte {

private String numero;

private String fechaEmision;

private Foto foto;

```
    private Titular titular;

    public Pasaporte(String numero, String fechaEmision, Foto foto, Titular
titular) {
        this.numero = numero;
        this.fechaEmision = fechaEmision;
        this.foto = foto;
        this.titular = titular;
        titular.setPasaporte(this); // bidireccional
    }

    // getters y setters
}

public class Foto {
    private String imagen;
    private String formato;

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

    // getters y setters
}

public class Titular {
    private String nombre;
    private String dni;
```

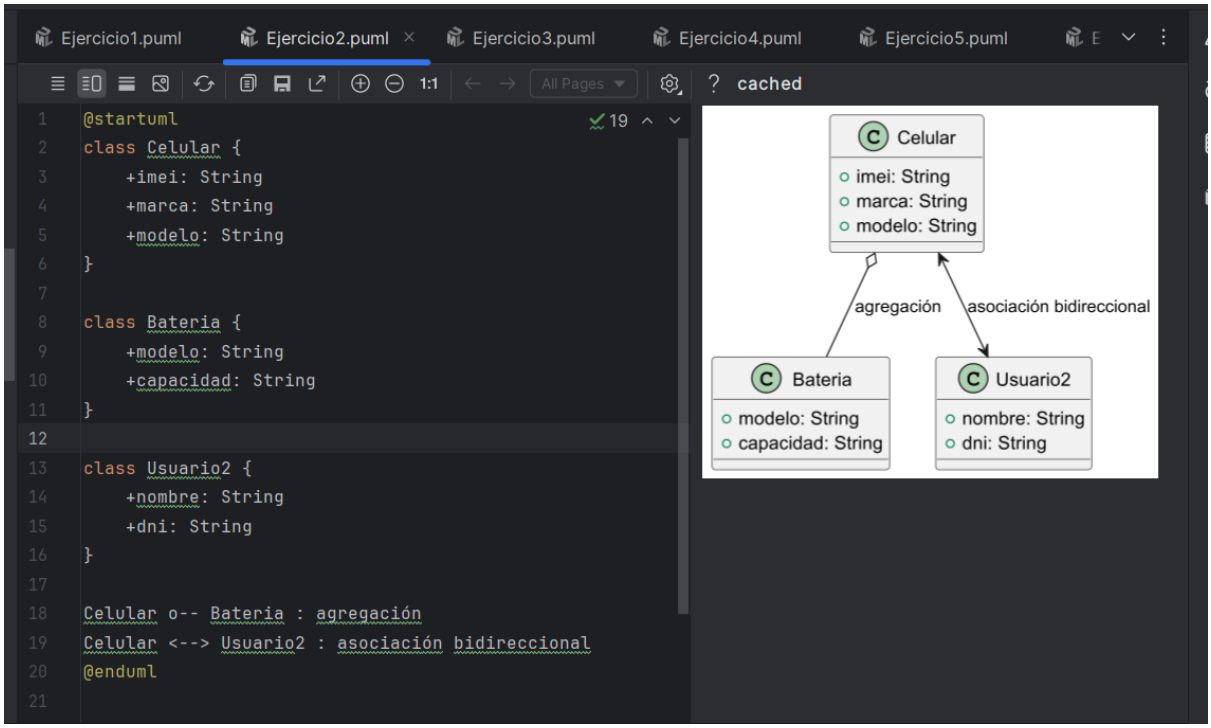
```
    private Pasaporte pasaporte;

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

    public void setPasaporte(Pasaporte pasaporte) {
        this.pasaporte = pasaporte;
    }

    // getters y setters
}
```

Explicación: Se implementó la composición con Foto dentro de Pasaporte, y la asociación bidireccional entre Pasaporte y Titular.

Ejercicio 2 – Celular, Batería, Usuario

Archivos: Celular.java, Bateria.java, Usuario.java

Diagrama UML con Su código:



Relaciones:

- Agregación: Celular → Batería
- Asociación bidireccional: Celular ↔ Usuario

Código Java: 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, Usuario usuario) {

this.imei = imei;

this.marca = marca;

this.modelo = modelo;

this.bateria = bateria;

this.usuario = usuario;

usuario.setCelular(this); // bidireccional

```
    }

    // getters y setters
}

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

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

    // getters y setters
}

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;
```

}

// getters y setters

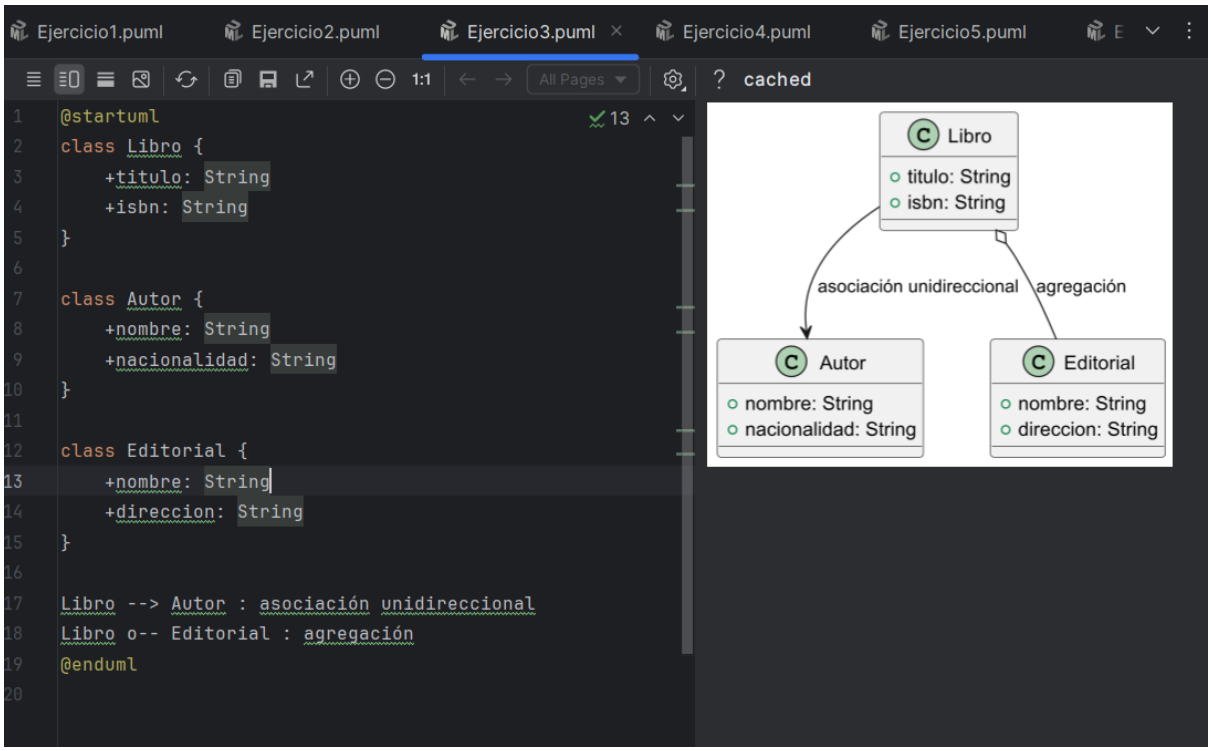
}

Explicación: La batería se agrega al celular (agregación), y se establece la asociación bidireccional entre celular y usuario.

Ejercicio 3 – Libro, Autor, Editorial

Archivos: Libro.java, Autor.java, Editorial.java

Diagrama UML con Su código:



Relaciones:

- Asociación unidireccional: Libro → Autor
- Agregación: Libro → Editorial

Código 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; // unidireccional
        this.editorial = editorial; // agregación
    }

    // getters y setters
}

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

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

    // getters y setters
}

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

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

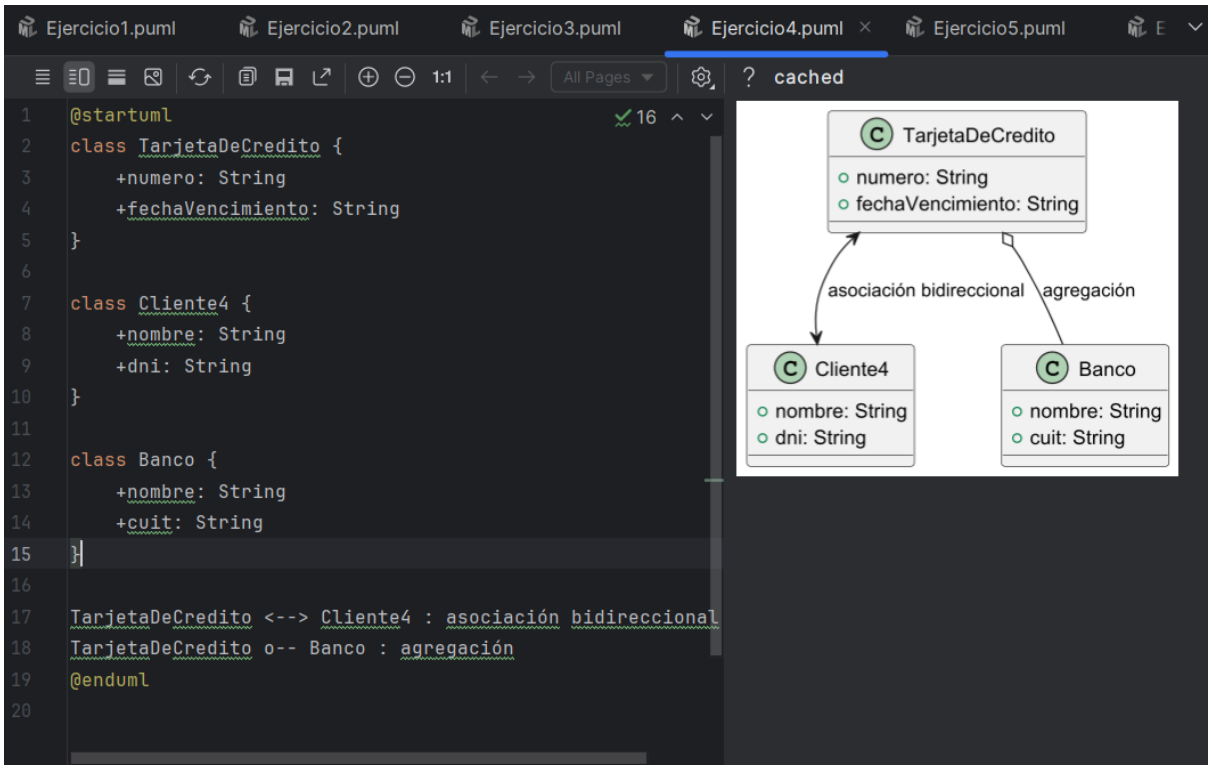
```
    }

    // getters y setters
}
```

Explicación: La relación con el autor es unidireccional, mientras que la editorial se agrega (agregación).

Ejercicio 4 – TarjetaDeCrédito, Cliente, Banco

Archivos: TarjetaDeCredito.java, Cliente4.java, Banco.java

Diagrama UML con Su código:



Relaciones:

- Asociación bidireccional: TarjetaDeCrédito ↔ Cliente
- Agregación: TarjetaDeCrédito → Banco

Codigo en java:

```
public class TarjetaDeCredito {
    private String numero;
    private String fechaVencimiento;
    private Cliente4 cliente;
```

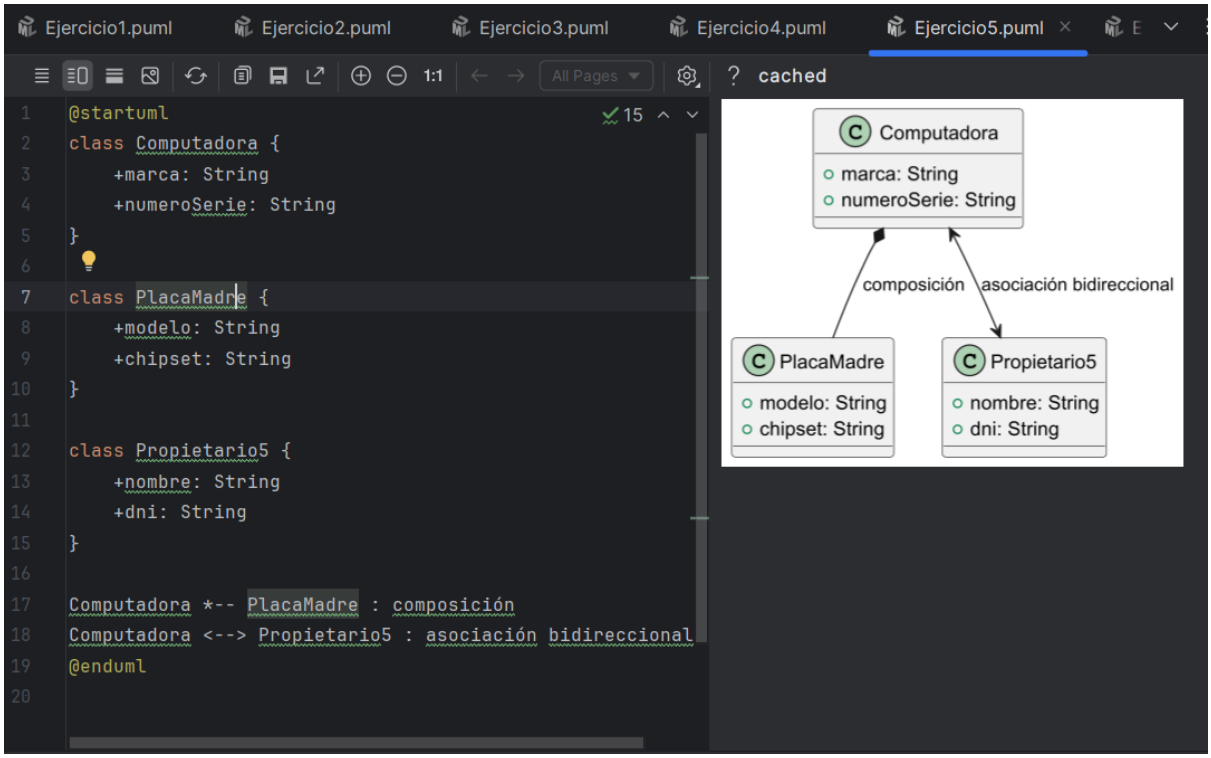
```
    private Banco banco;

    public TarjetaDeCredito(String numero, String fechaVencimiento, Cliente4 cliente,
Banco banco) {
        this.numero = numero;
        this.fechaVencimiento = fechaVencimiento;
        this.cliente = cliente;
        this.banco = banco;
    }
    // getters y setters
}

public class Cliente4 {
    private String nombre;
    private String dni;
    private TarjetaDeCredito tarjeta;

    public Cliente4(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setTarjeta(TarjetaDeCredito tarjeta) { this.tarjeta = tarjeta; }
    // getters
}

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

```
    public Banco(String nombre, String cuit) {
        this.nombre = nombre;
        this.cuit = cuit;
    }
    // getters y setters
}
```

Ejercicio 5 – Computadora, PlacaMadre, Propietario

Archivos: Computadora.java, PlacaMadre.java, Propietario5.java

Diagrama UML con su código:



Relaciones:

- Composición: Computadora → PlacaMadre
- Asociación bidireccional: Computadora ↔ Propietario

Código Java:

```
public class Computadora {
    private String marca;
    private String numeroSerie;
```

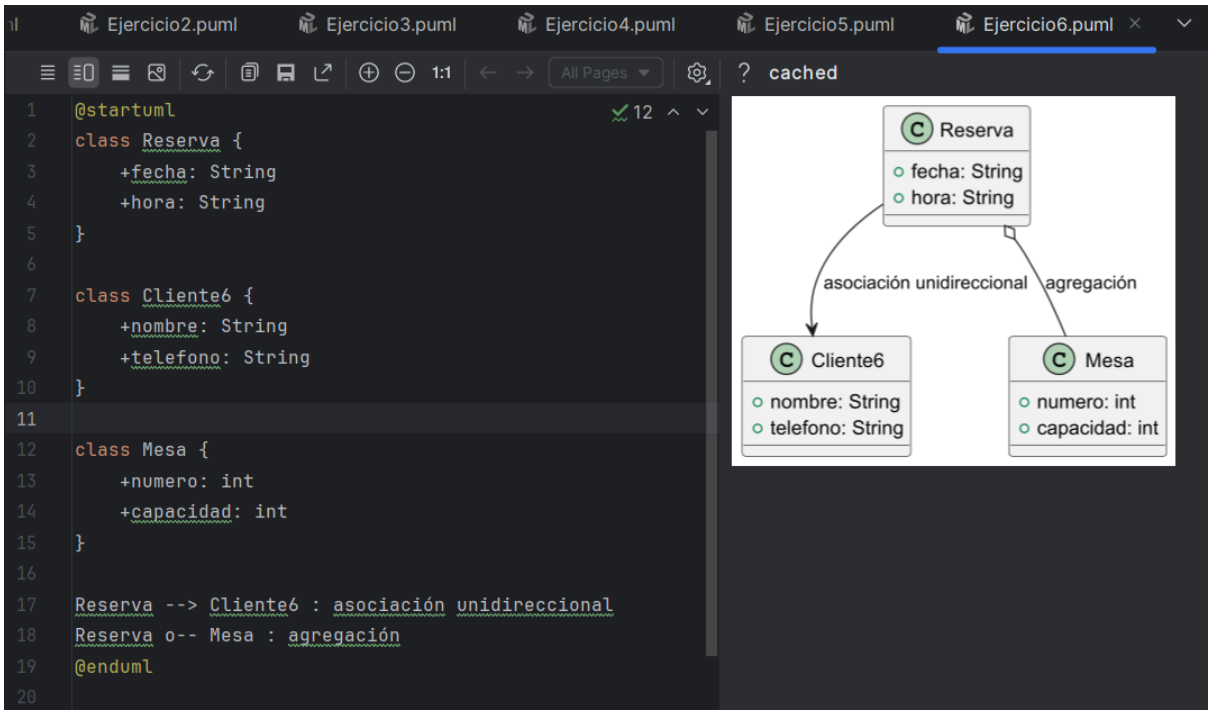
```
    private PlacaMadre placa;
    private Propietario5 propietario;

    public Computadora(String marca, String numeroSerie, PlacaMadre placa,
Propietario5 propietario) {
        this.marca = marca;
        this.numeroSerie = numeroSerie;
        this.placa = placa;
        this.propietario = propietario;
    }
    // getters y setters
}

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

    public PlacaMadre(String modelo, String chipset) {
        this.modelo = modelo;
        this.chipset = chipset;
    }
    // getters y setters
}

public class Propietario5 {
    private String nombre;
    private String dni;
    private Computadora computadora;
```

```
    public Propietario5(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }
    public void setComputadora(Computadora computadora) { this.computadora = computadora; }
    // getters
}
```

Ejercicio 6 – Reserva, Cliente, Mesa

Archivos: Reserva.java, Cliente6.java, Mesa.java

DIAGRAMA UML con su CODIGO:



Relaciones:

- Asociación unidireccional: Reserva → Cliente
- Agregación: Reserva → Mesa

Código Java:

```
public class Reserva {
    private String fecha;
    private String hora;
```

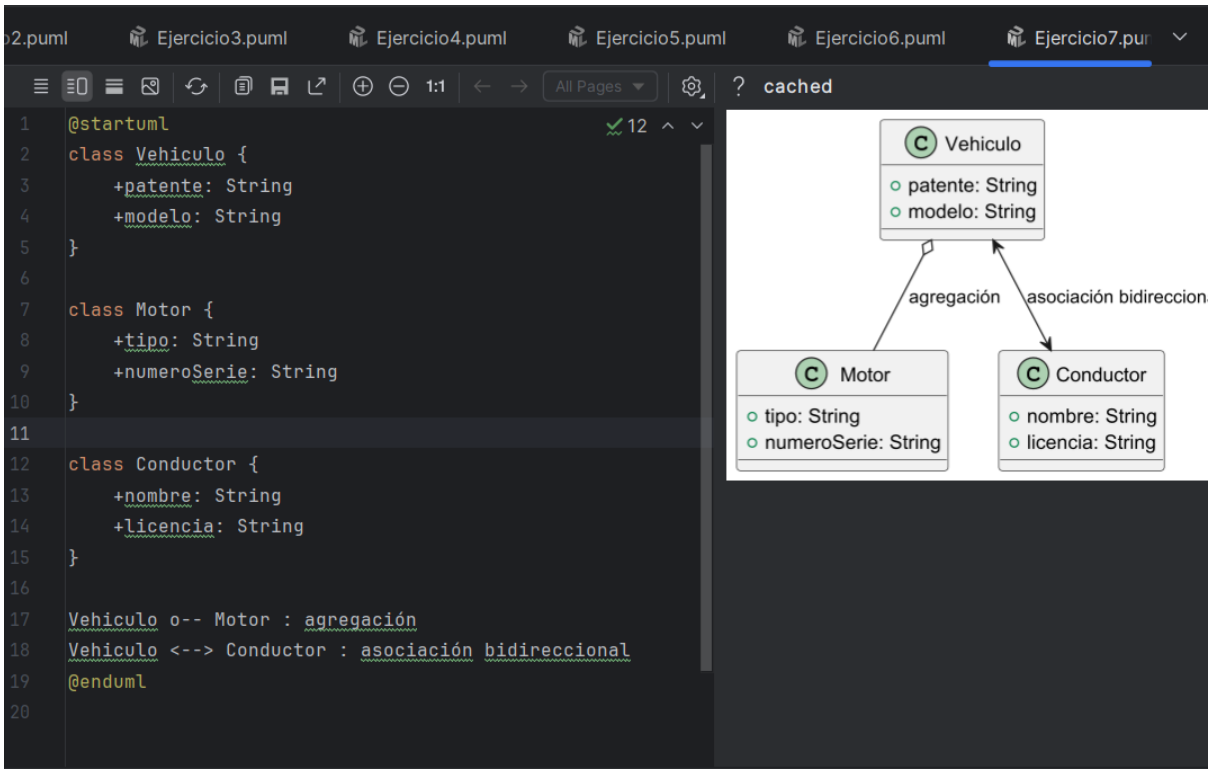
```
    private Cliente6 cliente;
    private Mesa mesa;

    public Reserva(String fecha, String hora, Cliente6 cliente, Mesa mesa) {
        this.fecha = fecha;
        this.hora = hora;
        this.cliente = cliente;
        this.mesa = mesa;
    }
    // getters y setters
}

public class Cliente6 {
    private String nombre;
    private String telefono;

    public Cliente6(String nombre, String telefono) {
        this.nombre = nombre;
        this.telefono = telefono;
    }
    // getters y setters
}

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

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

```
        this.capacidad = capacidad;
    }
    // getters y setters
}
```

Ejercicio 7 – Vehículo, Motor, Conductor

Archivos: Vehiculo.java, Motor.java, Conductor.java

DIAGRAMA UML con su CODIGO:



Relaciones:

- Agregación: Vehículo → Motor
- Asociación bidireccional: Vehículo ↔ Conductor

Código Java:

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

```
    public Vehiculo(String patente, String modelo, Motor motor, Conductor conductor)
{
        this.patente = patente;
        this.modelo = modelo;
        this.motor = motor;
        this.conductor = conductor;
    }
    // getters y setters
}

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

    public Motor(String tipo, String numeroSerie) {
        this.tipo = tipo;
        this.numeroSerie = numeroSerie;
    }
    // getters y setters
}

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

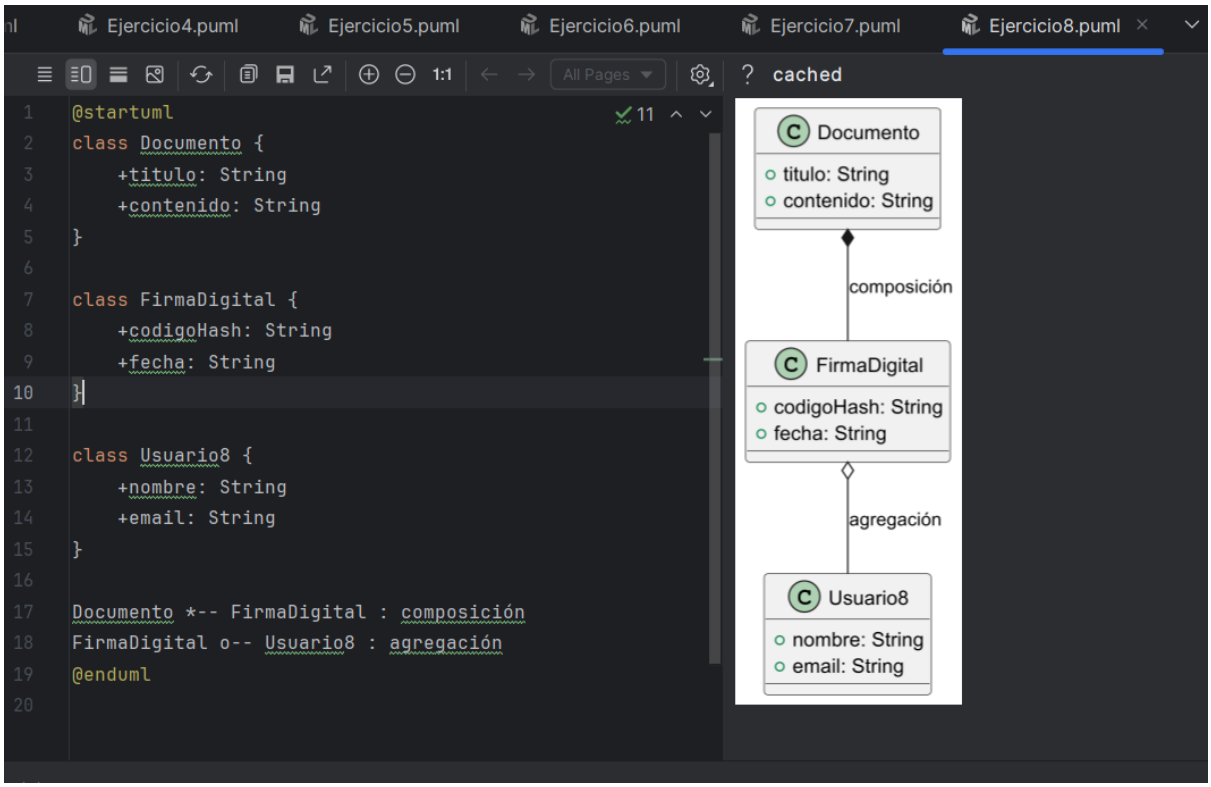
// getters

}

Ejercicio 8 – Documento, FirmaDigital, Usuario

Archivos: Documento.java, FirmaDigital.java, Usuario8.java

DIAGRAMA UML con su CODIGO:



Relaciones:

- Composición: Documento → FirmaDigital
- Agregación: FirmaDigital → Usuario

public class Documento {

private String titulo;

private String contenido;

private FirmaDigital firma;

public Documento(String titulo, String contenido, FirmaDigital firma) {

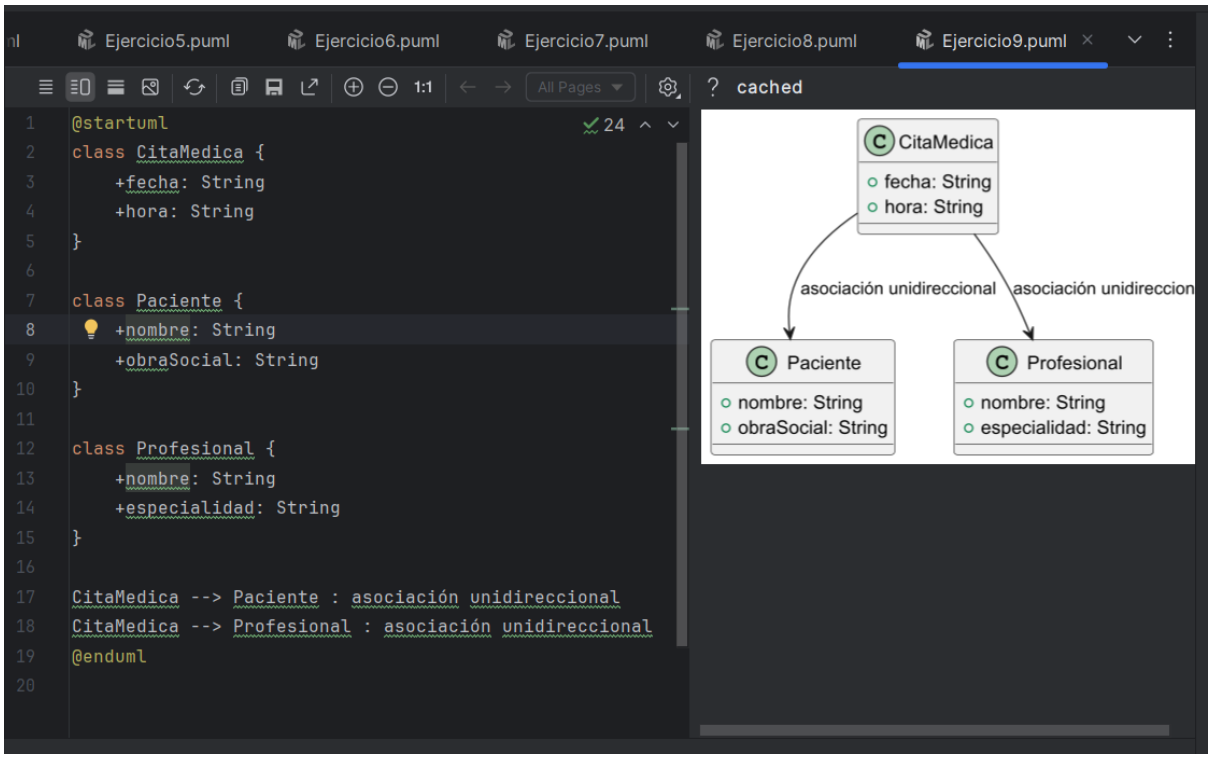
this.titulo = titulo;

```
        this.contenido = contenido;
        this.firma = firma;
    }
    // getters y setters
}

public class FirmaDigital {
    private String codigoHash;
    private String fecha;
    private Usuario8 usuario;

    public FirmaDigital(String codigoHash, String fecha, Usuario8 usuario) {
        this.codigoHash = codigoHash;
        this.fecha = fecha;
        this.usuario = usuario;
    }
    // getters y setters
}

public class Usuario8 {
    private String nombre;
    private String email;

    public Usuario8(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }
    // getters
}
```

Ejercicio 9 – CitaMédica, Paciente, Profesional

Archivos: CitaMedica.java, Paciente.java, Profesional.java

DIAGRAMA UML con su CODIGO:



Relaciones:

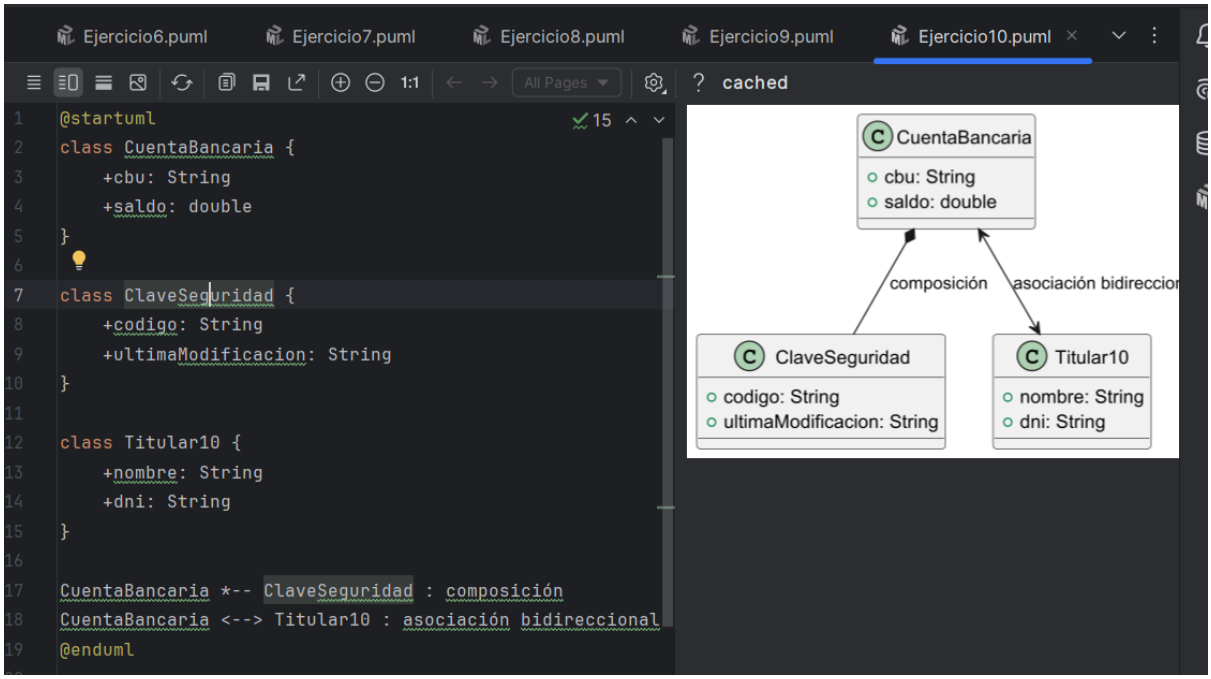
```
public class CitaMedica {
    private String fecha;
    private String hora;
    private Paciente paciente;
    private Profesional profesional;

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

```
    // getters y setters
}

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

    public Paciente(String nombre, String obraSocial) {
        this.nombre = nombre;
        this.obraSocial = obraSocial;
    }
    // getters y setters
}

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

    public Profesional(String nombre, String especialidad) {
        this.nombre = nombre;
        this.especialidad = especialidad;
    }
    // getters y setters
}
```

Ejercicio 10 – CuentaBancaria, ClaveSeguridad, Titular

Archivos: CuentaBancaria.java, ClaveSeguridad.java, Titular.java

DIAGRAMA UML con su CODIGO:



Relaciones:

- Composición: CuentaBancaria → ClaveSeguridad
- Asociación bidireccional: CuentaBancaria ↔ Titular

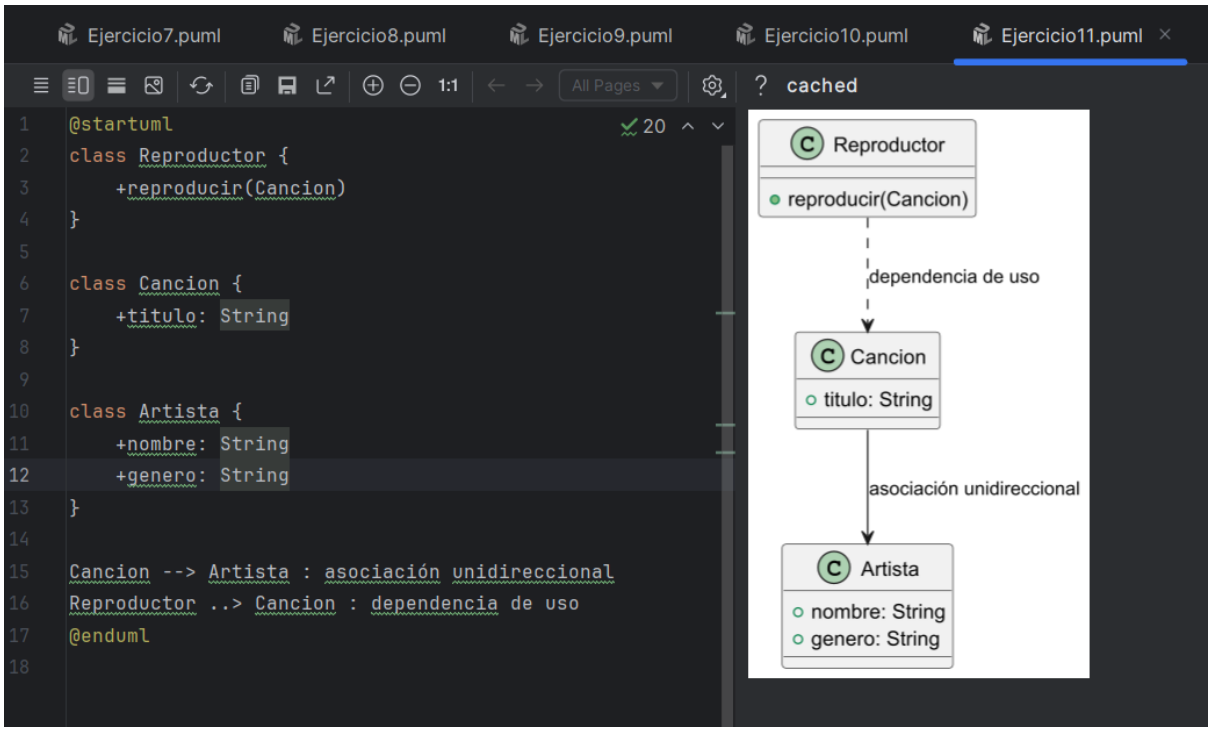
```
public class CuentaBancaria {
    private String cbu;
    private double saldo;
    private ClaveSeguridad clave;
    private Titular titular;

    public CuentaBancaria(String cbu, double saldo, ClaveSeguridad clave, Titular titular) {
        this.cbu = cbu;
        this.saldo = saldo;
        this.clave = clave;
        this.titular = titular;
    }
    // getters y setters
}
```

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

    public ClaveSeguridad(String codigo, String ultimaModificacion) {
        this.codigo = codigo;
        this.ultimaModificacion = ultimaModificacion;
    }
    // getters y setters
}

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; }
    // getters
}
```

jercicios de Dependencia de Uso

Ejercicio 11 – Reproductor, Canción, Artista

DIAGRAMA UML con su CODIGO:



Archivos: Reproductor.java, Cancion.java, Artista.java
Relaciones:

- Asociación unidireccional: Canción → Artista
- Dependencia de uso: Reproductor.reproducir(Cancion)

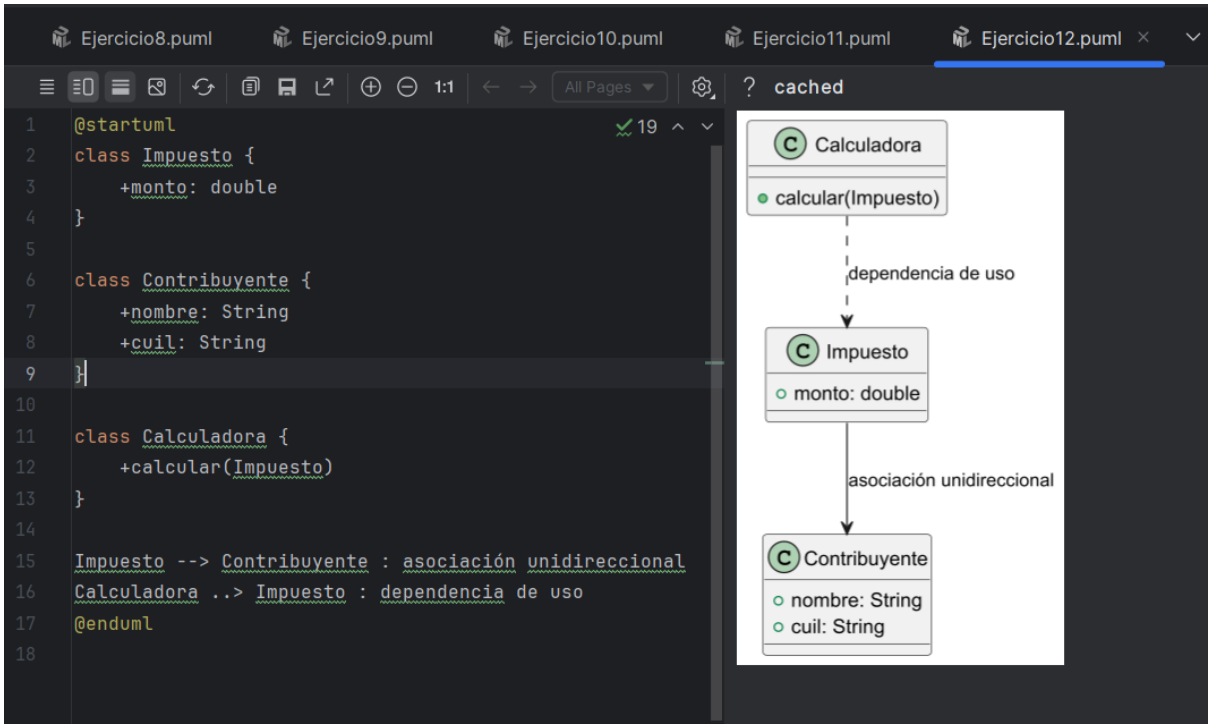
```
public class Reproductor {
    public void reproducir(Cancion cancion) {
        System.out.println("Reproduciendo: " + cancion.getTitulo());
    }
}


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; }
    // getters y setters
}

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

    public Artista(String nombre, String genero) {
        this.nombre = nombre;
        this.genero = genero;
    }
    // getters y setters
}
```

Ejercicio 12 – Impuesto, Contribuyente, Calculadora

Archivos: Impuesto.java, Contribuyente.java, Calculadora.java

DIAGRAMA UML con su CODIGO:



Relaciones:

- Asociación unidireccional: Impuesto → Contribuyente
- Dependencia de uso: Calculadora.calcular(Impuesto)

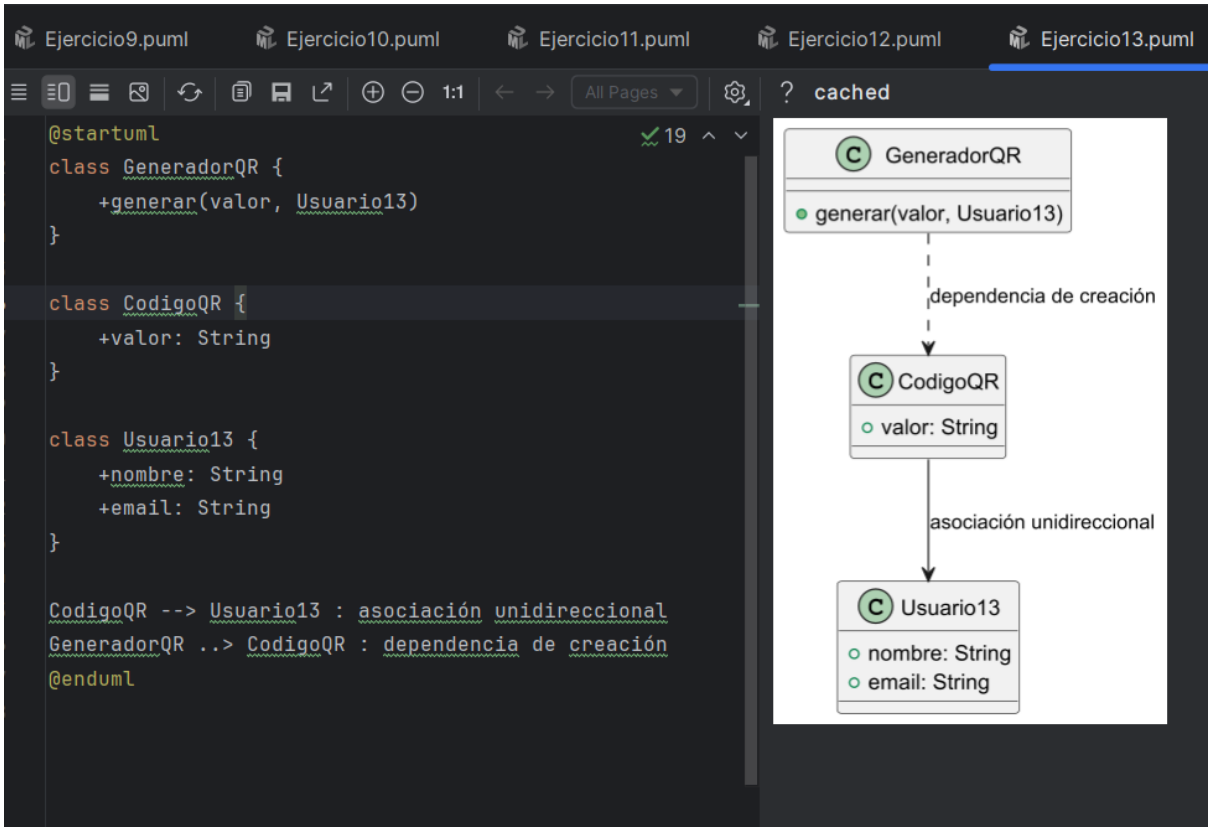
```
public class Calculadora {
    public void calcular(Impuesto impuesto) {
        System.out.println("Calculando impuesto: " + impuesto.getMonto());
    }
}


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; }
```

```
    // getters y setters
}

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

    public Contribuyente(String nombre, String cuil) {
        this.nombre = nombre;
        this.cuil = cuil;
    }
    // getters y setters
}
```

Ejercicios de Dependencia de Creación

Ejercicio 13 – GeneradorQR, Usuario, CodigoQR

DIAGRAMA UML con su CODIGO:



Archivos: GeneradorQR.java, Usuario.java, CodigoQR.java
Relaciones:

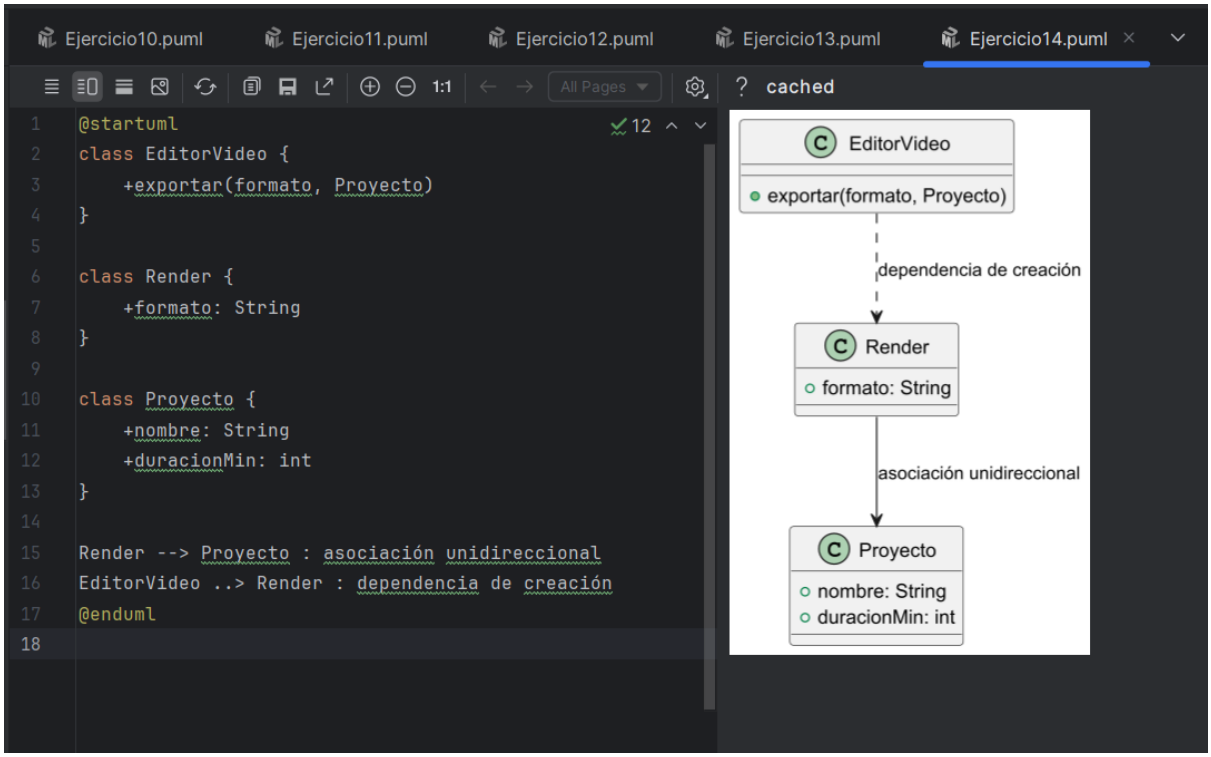
- Asociación unidireccional: CodigoQR → Usuario
- Dependencia de creación: GeneradorQR.generar(String, Usuario)

```
public class GeneradorQR {
    public CodigoQR generar(String valor, Usuario usuario) {
        return new CodigoQR(valor, usuario);
    }
}

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

    public CodigoQR(String valor, Usuario usuario) {
        this.valor = valor;
        this.usuario = usuario;
    }
    // getters y setters
}

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

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

```
    // getters
}
```

Ejercicio 14 – EditorVideo, Proyecto, Render

DIAGRAMA UML con su CODIGO:



Archivos: EditorVideo.java, Proyecto.java, Render.java
Relaciones:

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