

PROGRAMACIÓN II

Trabajo Práctico 5: Relaciones UML 1 a 1

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.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Asociación	Relación entre clases con referencia mutua o directa, puede ser uni o bidireccional
Agregación	Relación de "tiene un" donde los objetos pueden vivir independientemente
Composición	Relación fuerte de contención, el ciclo de vida del objeto contenido depende del otro
Dependencia de uso	Una clase usa otra como parámetro en un método, sin almacenarla como atributo
Dependencia de creación	Una clase crea otra en tiempo de ejecución, sin mantenerla como atributo
Asociación	Relación entre clases con referencia mutua o directa, puede ser uni o bidireccional
Agregación	Relación de "tiene un" donde los objetos pueden vivir independientemente

url repositorio git hub

https://github.com/Wally-ux/UTN_Programacion_2/tree/main/5_UML

Caso Práctico

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

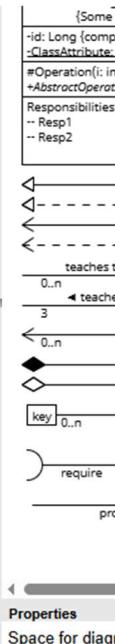
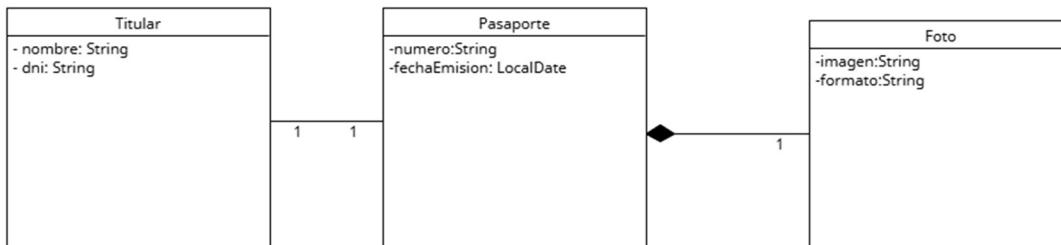
- Diagrama UML
- Tipo de relación (asociación, agregación, composición, dependencia)
- Dirección (unidireccional o bidireccional)
- Implementación de las clases con atributos y relaciones definidas

Ejercicios de Relaciones 1 a 1

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

Clases y atributos:

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



iv.

Composicion fuerte →pasaporte y foto

v.

Vemos que foto no puede existir sin pasaporte, si eliminamos pasaporte tambien se elimina foto
El todo controla la parte dependiente, pasaporte requiere una foto



Pasaporte – titular

Tenemos un vínculo donde la asociación donde nadie contiene a nadie en este caso.

Si destruimos el objeto pasaporte, el objeto titular sigue existiendo, la cardinalidad 1-1 indica que cada pasaporte tiene un titular y cada titular tiene un pasaporte.

The screenshot shows a Java code editor with four tabs at the top: Main.java [-/A] ×, Titular.java [-/A] × (which is the active tab), Pasaporte.java [-/A] ×, and Foto.java [-/A] ×. The Titular.java code is as follows:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to ed
4   */
5 package pkg1_pasaporte;
6
7
8 public class Titular {
9     private String nombre;
10    private String dni;
11    private Pasaporte pasaporte;
12
13    public Titular(String nombre, String dni) {
14        this.nombre = nombre;
15        this.dni = dni;
16    }
17
18    void setPasaporte(Pasaporte pas) {
19        this.pasaporte = pas; }
20
21    public Pasaporte getPasaporte() {
22        return pasaporte; }
23    // getters/setters básicos
24
25 }
```

```
public class Pasaporte {  
    private String numero;  
    private String fechaEmision;  
    private Foto foto;  
    // Asociación bi  
    private Titular titular;  
  
    private Pasaporte(String numero, String fechaEmision, Foto foto, Titular titular) {  
        this.numero = numero;  
        this.fechaEmision = fechaEmision;  
        this.foto = foto;  
    }  
  
    public String getNumero() {  
        return numero;  
    }  
  
    public void setNumero(String numero) {  
        this.numero = numero;  
    }  
  
    public String getFechaEmision() {  
        return fechaEmision;  
    }
```

```
public void setFechaEmision(String fechaEmision) {  
    this.fechaEmision = fechaEmision;  
}  
  
public Foto getFoto() {  
    return foto;  
}  
  
public void setFoto(Foto foto) {  
    this.foto = foto;  
}  
  
public Titular getTitular() {  
    return titular;  
}  
  
public void setTitular(Titular titular) {  
    this.titular = titular;  
}
```

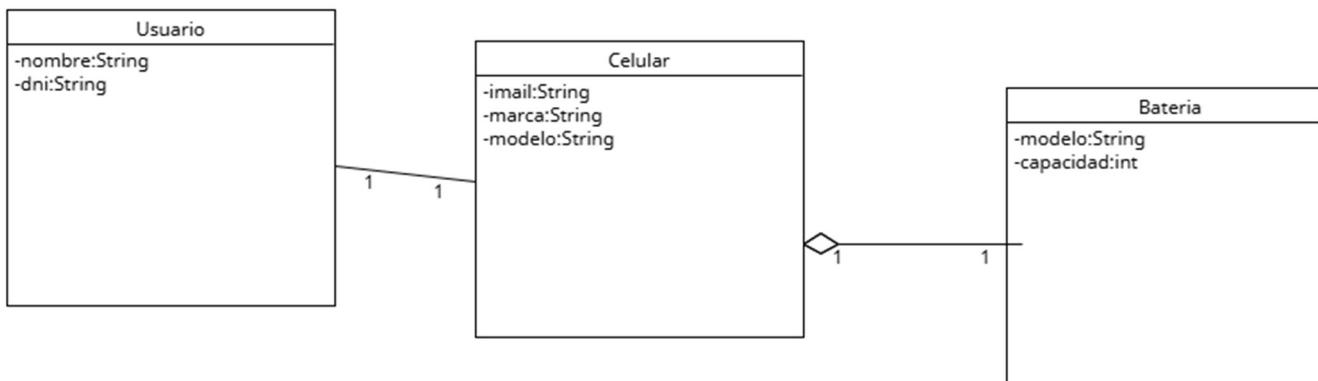


2. Celular - Batería - Usuario

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

Clases y atributos:

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



Celular –bateria

Agregacion unidireccional , el celular tiene una bateria la cual es independiente del celular,la bateria puede existir o reemplazarse independiente del celular, por lo que tiene una relacion débil

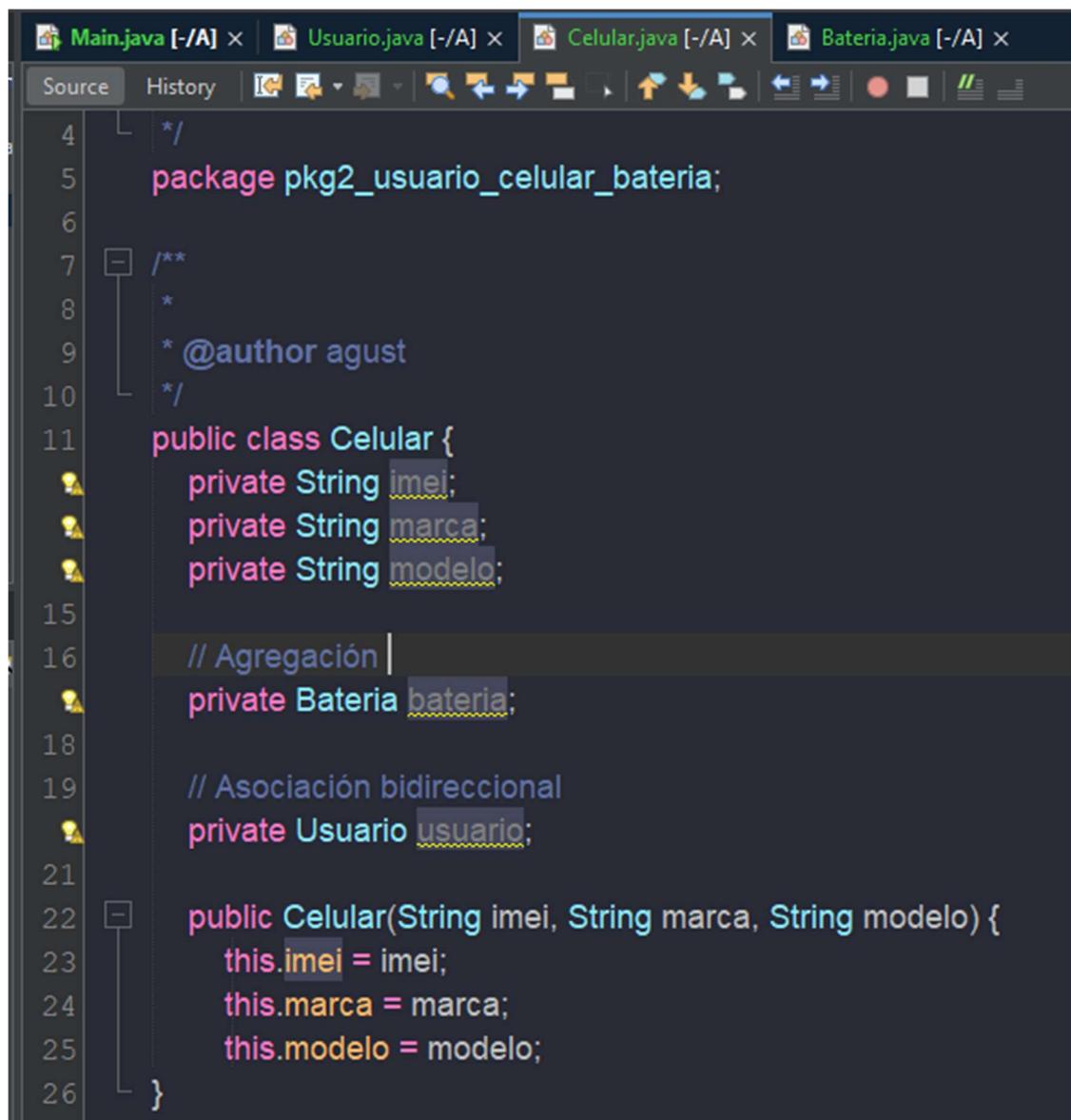
Celular - Usuario

Entre el celular y el usuario ninguno contiene al otro.



The screenshot shows a Java code editor with four tabs at the top: Main.java, Usuario.java, Celular.java, and Bateria.java. The Usuario.java tab is active. The code defines a class Usuario with private attributes nombre, dni, and celular, and methods getNombre(), getDni(), getCelular(), setNombre(), setDni(), and setCelular(). The code is color-coded, and line numbers are visible on the left.

```
4 package pkg2_usuario_celular_bateria;
5
6 /**
7 * @author agust
8 */
9
10 public class Usuario {
11     private String nombre;
12     private String dni;
13     private Celular celular; //encale con la otra clase
14
15
16     public Usuario(String nombre, String dni) {
17         this.nombre = nombre;
18         this.dni = dni;
19     }
20
21     public String getNombre() {
22         return nombre;
23     }
24     public String getDni() {
25         return dni;
26     }
27     public Celular getCelular() {
28         return celular;
29     }
30
31     public void setNombre(String nombre) {
32         this.nombre = nombre;
33     }
34
35     public void setDni(String dni) {
36         this.dni = dni;
37     }
38
39     public void setCelular(Celular celular) {
40         this.celular = celular;
41     }
42
43
44 }
45
```



```
4  */
5  package pkg2_usuario_celular_bateria;
6
7  /**
8  *
9  * @author agust
10 */
11 public class Celular {
12     private String imei;
13     private String marca;
14     private String modelo;
15
16     // Agregación |
17     private Bateria bateria;
18
19     // Asociación bidireccional
20     private Usuario usuario;
21
22     public Celular(String imei, String marca, String modelo) {
23         this.imei = imei;
24         this.marca = marca;
25         this.modelo = modelo;
26     }
}
```



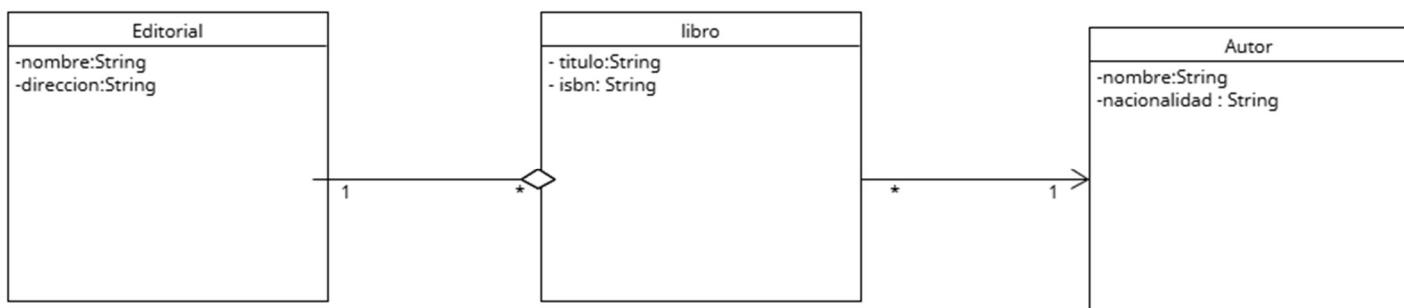
```
5 package pkg2_usuario_celular_bateria;
6
7 /**
8 *
9 * @author agust
10 */
11 public class Bateria {
12     private String modelo;
13     private int capacidad;
14
15     public Bateria(String modelo, int capacidad) {
16         this.modelo = modelo;
17         this.capacidad = capacidad;
18     }
19
20     public String getModelo() {
21         return modelo; }
22     public int getCapacidad() {
23         return capacidad;
24     }
25
26     public void setModelo(String modelo) {
27         this.modelo = modelo;
28     }
29
30     public void setCapacidad(int capacidad) {
31         this.capacidad = capacidad;
32     }
33 }
```

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

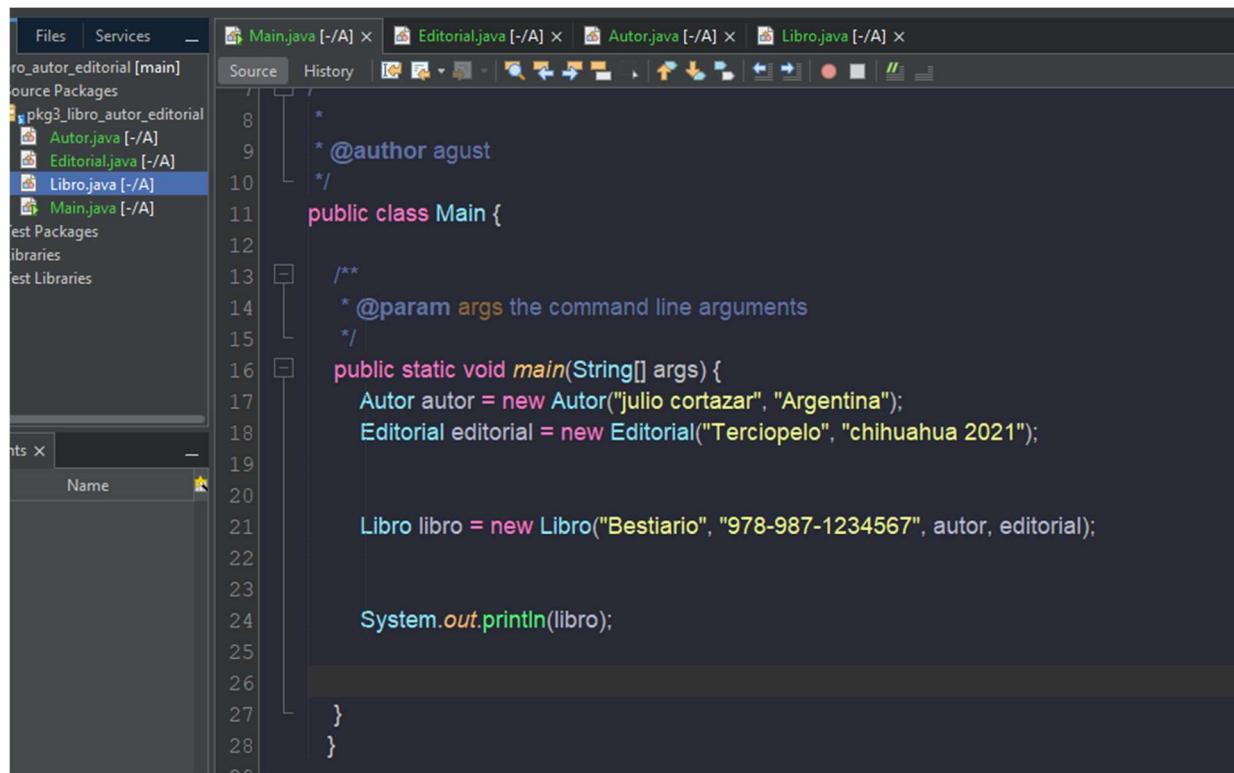
Clases y atributos:

- i. Libro: titulo, isbn
- ii. Autor: nombre, nacionalidad
- iii. Editorial: nombre, direccion

Modelo uml:



Clase main



```
>Main.java [-/A] x  Editorial.java [-/A] x  Autor.java [-/A] x  Libro.java [-/A] x
Source History
8  *
9  * @author agust
10 */
11 public class Main {
12 /**
13  * @param args the command line arguments
14 */
15 public static void main(String[] args) {
16     Autor autor = new Autor("julio cortazar", "Argentina");
17     Editorial editorial = new Editorial("Terciopelo", "chihuahua 2021");
18
19     Libro libro = new Libro("Bestiario", "978-987-1234567", autor, editorial);
20
21     System.out.println(libro);
22
23 }
24 }
25
26 }
27 }
28 }
```

Clase editorial

```
8      * @author agust
9
10     */
11
12    public class Editorial {
13        private String nombre;
14        private String direccion;
15
16        public Editorial (String nombre , String direccion) {
17            this.nombre =nombre;
18            this.direccion= direccion;
19        }
20
21        public String getNombre() {
22            return nombre;
23        }
24
25        public String getDireccion() {
26            return direccion;
27        }
28
```

```
29
30    @Override
31    public String toString() {
32        return "Editorial{" + "nombre=" + nombre + ", direccion=" + direccion + '}';
33    }
34
35}
```

Clase Autor



The screenshot shows the code for the **Autor** class in a Java IDE. The code includes constructor parameters, getter methods, and an overridden `toString()` method.

```
10  */
11 public class Autor {
12     private String nombre;
13     private String nacionalidad;
14
15     public Autor (String nombre , String nacionalidad){
16         this.nombre = nombre;
17         this.nacionalidad = nacionalidad;
18     }
19
20     public String getNombre() {
21         return nombre;
22     }
23
24     public String getNacionalidad() {
25         return nacionalidad;
26     }
27
28     @Override
29     public String toString() {
30         return "Autor{" + "nombre=" + nombre + ", nacionalidad=" + nacionalidad + '}';
31     }
32
33 }
```

Clase Libro

The screenshot shows the code for the **Libro** class in a Java IDE. The code includes methods to get the author and editorial, and an overridden `toString()` method.

```
33 }
34
35     public Autor getAutor() {
36         return autor;
37     }
38
39     public Editorial getEditorial() {
40         return editorial;
41     }
42
43     @Override
44     public String toString() {
45         return "Libro{" + "titulo= "
46             + titulo + ", isbn="
47             + isbn + ", autor="
48             + autor + ", editorial="
49             + editorial + '}';
50     }
51
52 }
53 }
```

```
Output - 3_libro_autor_editorial (run) ×

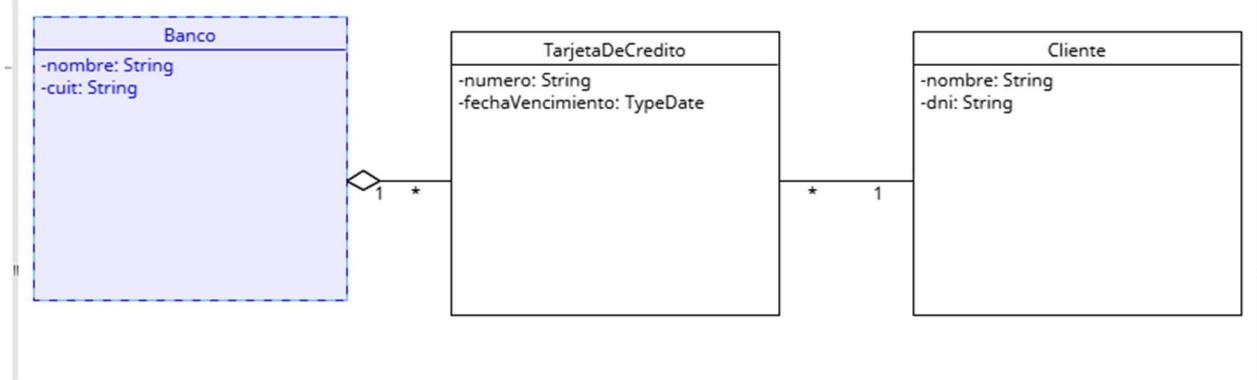
▶ run:
▶ Libro{
  titulo= Bestiario
  isbn=978-987-1234567
  autor=Autor{nombre=Argentina, nacionalidad=null}
  editorial=Editorial{
    nombre=Terciopelo
    , direccion=chihuahua 2021}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

4.TarjetaDeCrédito - Cliente - Banco

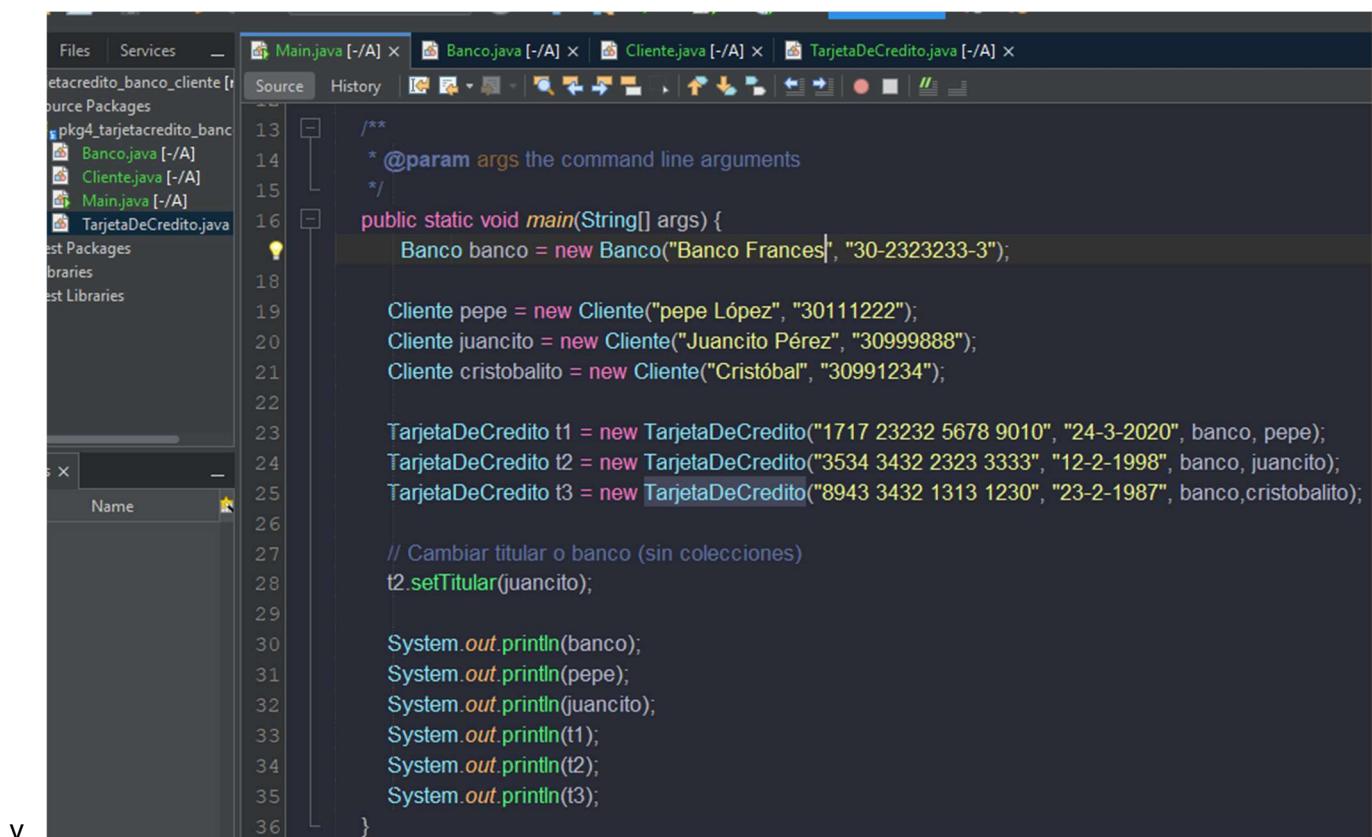
- c. Asociación bidireccional: **TarjetaDeCrédito ↔ Cliente**
- d. Agregación: **TarjetaDeCrédito → Banco**

Clases y atributos:

- i. TarjetaDeCrédito: numero, fechaVencimiento
- ii. Cliente: nombre, dni
- iii. Banco: nombre, cuit

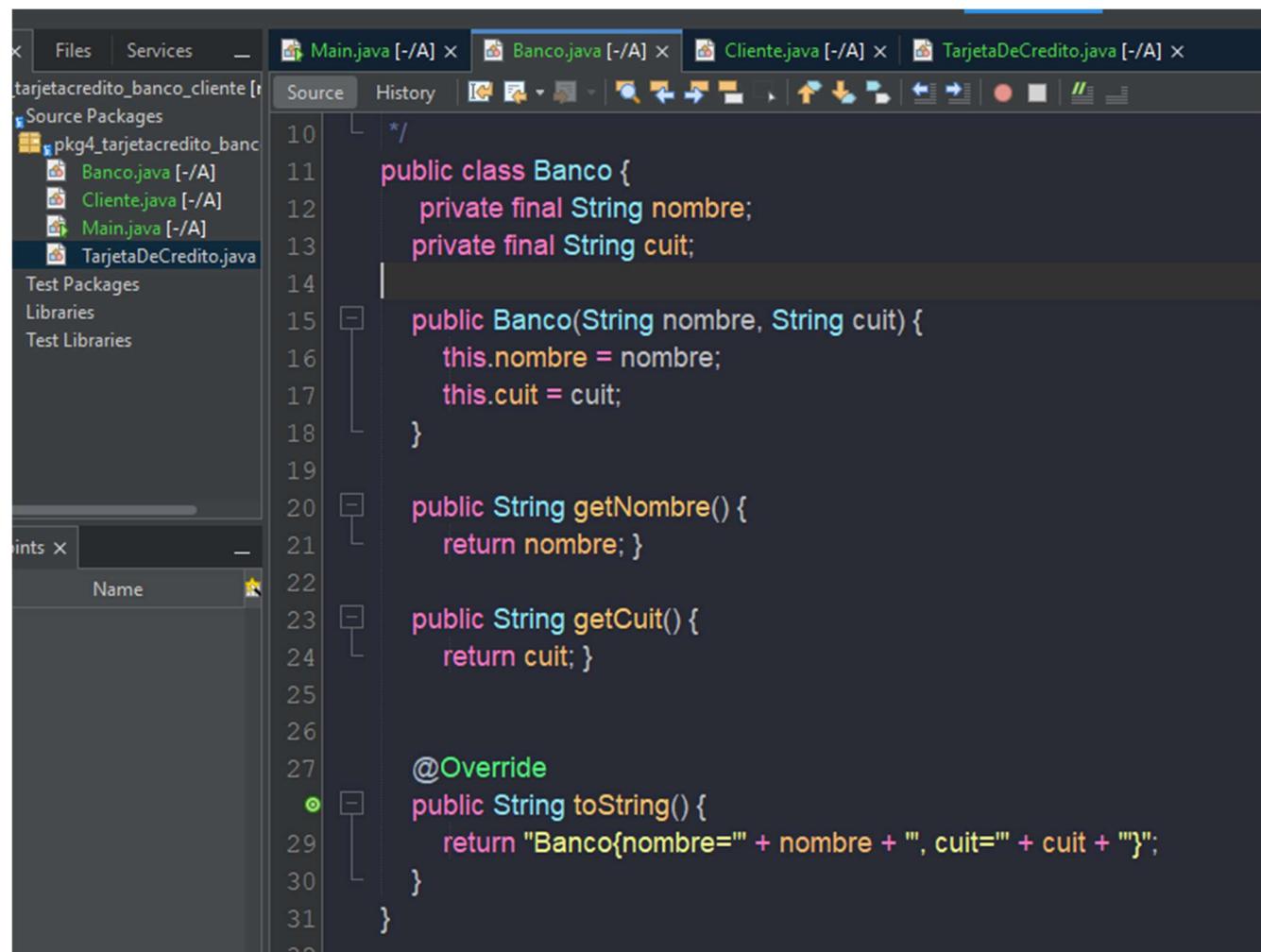


iv.



The screenshot shows a Java code editor with the following code:

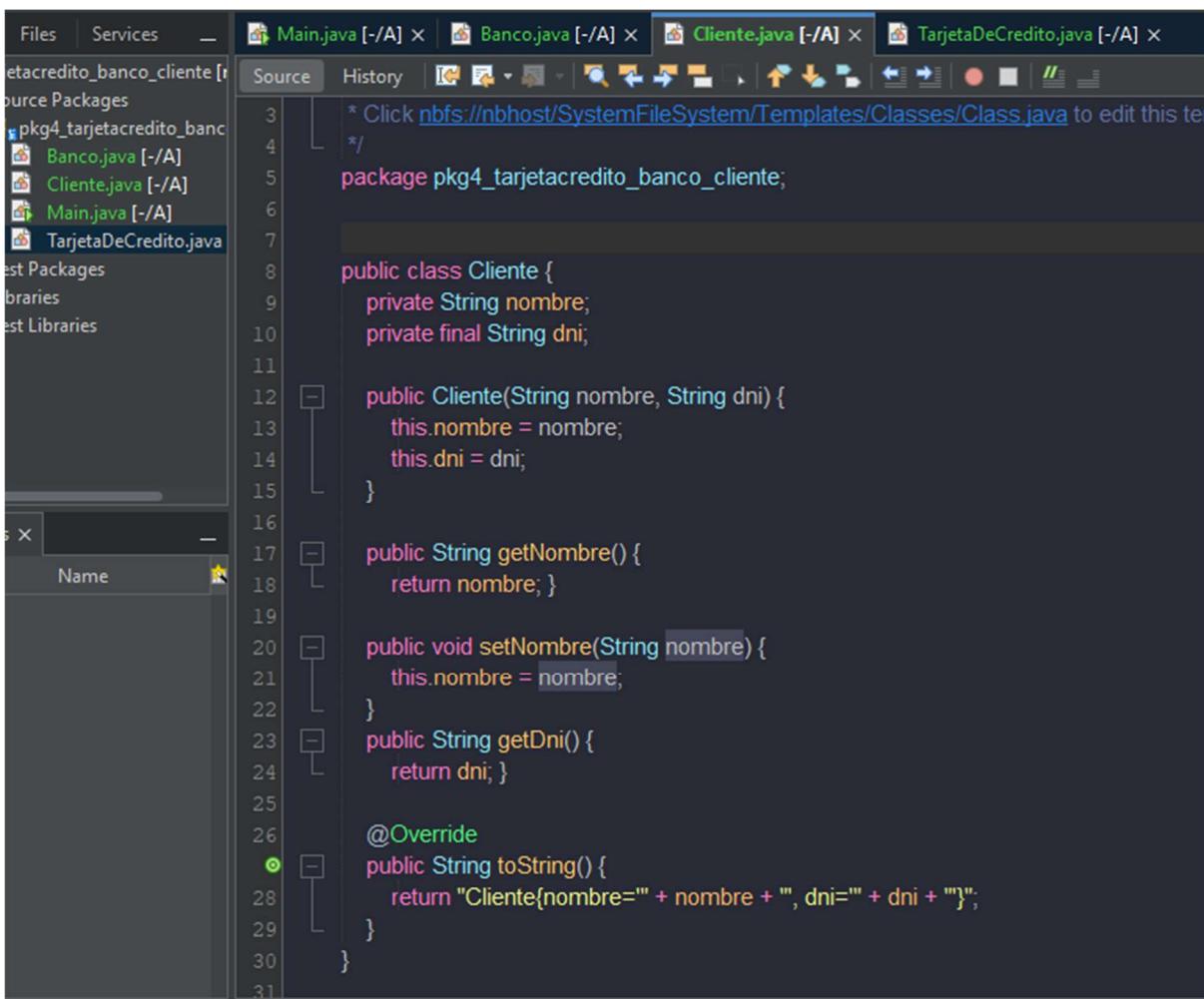
```
13  /**
14   * @param args the command line arguments
15  */
16 public static void main(String[] args) {
17     Banco banco = new Banco("Banco Frances", "30-2323233-3");
18
19     Cliente pepe = new Cliente("pepe López", "30111222");
20     Cliente juancito = new Cliente("Juancito Pérez", "30999888");
21     Cliente cristobalito = new Cliente("Cristóbal", "30991234");
22
23     TarjetaDeCredito t1 = new TarjetaDeCredito("1717 2323 5678 9010", "24-3-2020", banco, pepe);
24     TarjetaDeCredito t2 = new TarjetaDeCredito("3534 3432 2323 3333", "12-2-1998", banco, juancito);
25     TarjetaDeCredito t3 = new TarjetaDeCredito("8943 3432 1313 1230", "23-2-1987", banco,cristobalito);
26
27     // Cambiar titular o banco (sin colecciones)
28     t2.setTitular(juancito);
29
30     System.out.println(banco);
31     System.out.println(pepe);
32     System.out.println(juancito);
33     System.out.println(t1);
34     System.out.println(t2);
35     System.out.println(t3);
36 }
```



The screenshot shows a Java code editor with the following code:

```
10  * /  
11  public class Banco {  
12      private final String nombre;  
13      private final String cuit;  
14  
15      public Banco(String nombre, String cuit) {  
16          this.nombre = nombre;  
17          this.cuit = cuit;  
18      }  
19  
20      public String getNombre() {  
21          return nombre; }  
22  
23      public String getCuit() {  
24          return cuit; }  
25  
26  
27      @Override  
28      public String toString() {  
29          return "Banco{" + nombre + ", cuit=" + cuit + "}";  
30      }  
31  }  
32 }
```

vii.



```

package pkg4_tarjetacredito_banco_cliente;

public class Cliente {
    private String nombre;
    private final String dni;

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

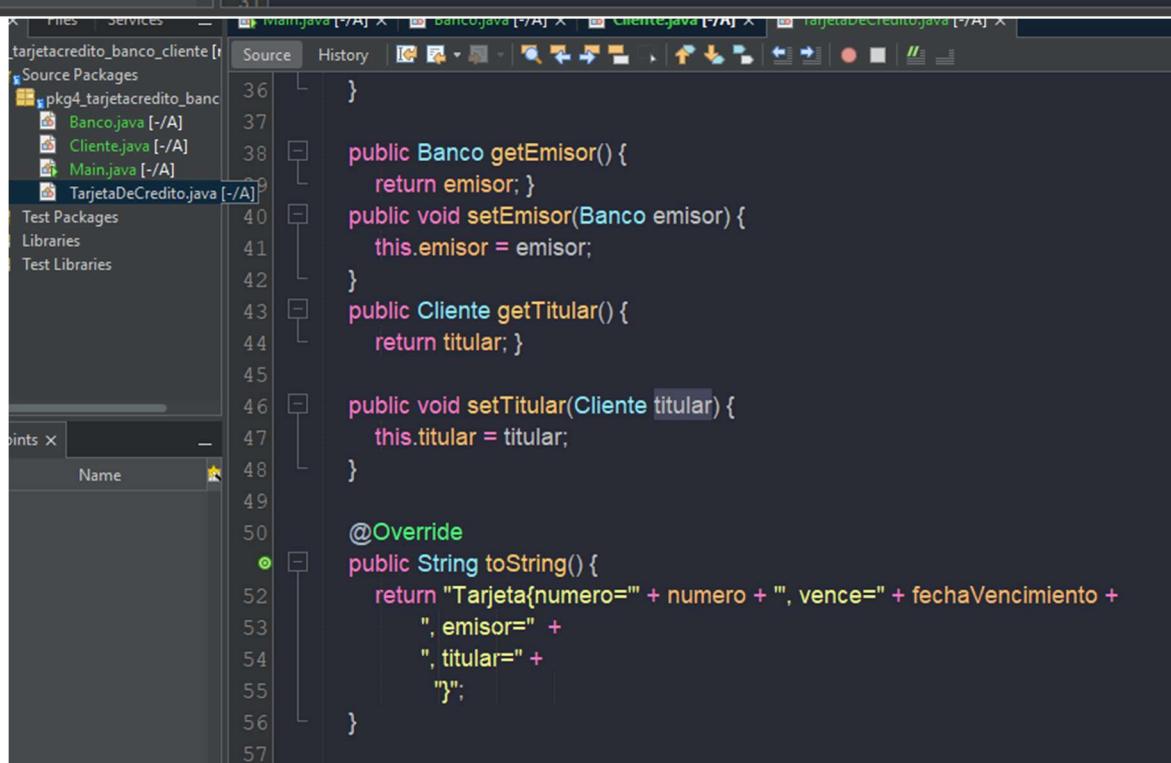
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getDni() {
        return dni;
    }

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

```

viii.



```

public Banco getEmisor() {
    return emisor;
}

public void setEmisor(Banco emisor) {
    this.emisor = emisor;
}

public Cliente getTitular() {
    return titular;
}

public void setTitular(Cliente titular) {
    this.titular = titular;
}

@Override
public String toString() {
    return "Tarjeta{" + "numero=" + numero + ", vence=" + fechaVencimiento +
        ", emisor=" +
        ", titular=" +
        "}";
}

```

ix. Impresion por consola

```
Output X
UTN_PROGRAMACION_II_TP - C:\Users\agust\OneDrive\Documentos\NetBeansProjects\UTN_PROGRAMACION_II_TP × 4_tarjetacredito_banco_cliente (run) ×

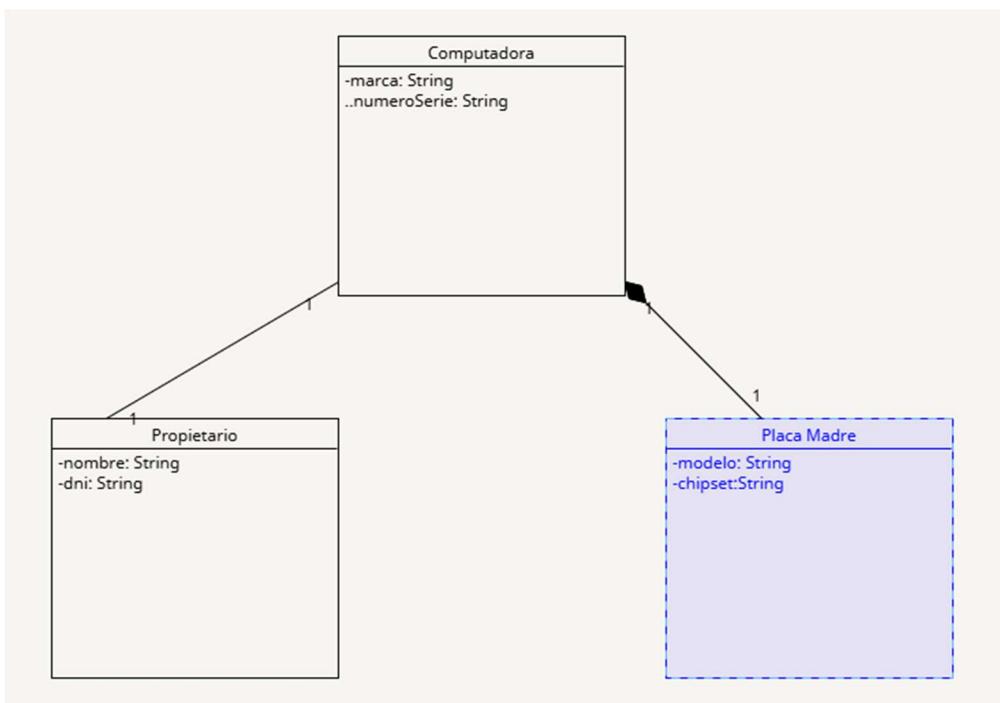
run:
Banco{nombre='Banco Frances', cuit='30-2323233-3'}
Cliente{nombre='pepe Lopez', dni='30111222'}
Cliente{nombre='Juancito Perez', dni='30999888'}
Tarjeta{numero='1717 23232 5678 9010', vence=24-3-2020, emisor=, titular=}
Tarjeta{numero='3534 3432 2323 3333', vence=12-2-1998, emisor=, titular=}
Tarjeta{numero='8943 3432 1313 1230', vence=23-2-1987, emisor=, titular=}
BUILD SUCCESSFUL (total time: 0 seconds)
```

X.

5-Computadora - PlacaMadre - Propietario

e. Composición: **Computadora → PlacaMadre**

f. Asociación bidireccional: **Computadora ↔ Propietario**



g.

Clases y atributos:

i. Computadora: marca, numeroSerie

```
4
5     package pkg5_computadora_placa_propietario;
6
7     /**
8      *
9      * @author agust
10     */
11    public class Computadora {
12        private String marca;
13        private String numeroSerie;
14        private PlacaMadre placaMadre; //aca tenemos la comosicion
15        private Propietario propietario; //aca tenemos la asociacion bidireccional
16
17        public Computadora(String marca, String numeroSerie, String modelo, String chipset) {
18            this.marca = marca;
19            this.numeroSerie = numeroSerie;
20            this.placaMadre = new PlacaMadre(modelo, chipset);
21            //dentro del constructor computadora se creas internamente su placa madre
22            //si se destruye la pc, la placa no tiene sentido sola
23        }
24    }
```

ii. PlacaMadre: modelo, chipset

```
4  //  
5  package pkg5_computadora_placa_propietario;  
6  
7  /**  
8  *  
9  * @author agust  
*/  
10 public class PlacaMadre {  
11     private String modelo;  
12     private String chipset;  
13     //constructor de placa madre  
14     public PlacaMadre(String modelo, String chipset) {  
15         this.modelo = modelo;  
16         this.chipset = chipset;  
17     }  
18     //getters and setters  
19     public String getModelo() {  
20         return modelo;  
21     }  
22  
23     public void setModelo(String modelo) {  
24         this.modelo = modelo;  
25     }  
26  
27     public String getChipset() {  
28         return chipset;  
29     }  
30     //metodo public para que sea llamda de otra clase no devuelve nada  
31     //recibe el parametro chipset y asigna el valor al atributo interno del objeto  
32     public void setChipset(String chipset) {  
33         this.chipset = chipset;  
34     }  
35 }  
36  
37 }
```

iii. Propietario: nombre, dni



```
10   */
11  public class Propietario {
12      private String nombre;
13      private String dni;
14      private Computadora computadora; //aociacion bidireccional
15
16      //constructor con todos los atributos de la clase
17      public Propietario(String nombre, String dni) {
18          this.nombre = nombre;
19          this.dni = dni;
20      }
21      //getters & setters
22      public String getNombre() {
23          return nombre;
24      }
25
26      public void setNombre(String nombre) {
27          this.nombre = nombre;
28      }
29
30      public String getDni() {
31          return dni;
32      }
33
34      public void setDni(String dni) {
35          this.dni = dni;
36      }
37
38      public Computadora getComputadora() {
39          return computadora;
40      }
41      //si computadora conoce al propietario , el propietario tambien conoce a la computadora
42      public void setComputadora(Computadora computadora) {
43          this.computadora = computadora;
44          //evitamos el bucle infinito
45          if (computadora != null && computadora.getPropietario() != this) {
46              computadora.setPropietario(this);
47          }
48      }
49
50
51  }
```



```
Main
10
11  public class Main {
12
13      /**
14      * @param args the command line arguments
15      */
16
17      public static void main(String[] args) {
18          // Creamos un propietario
19          Propietario propietario1 = new Propietario("Jhoony Depp ", "12345678");
20
21          // creamos una computadora con su placa madre (composición)
22          Computadora pc1 = new Computadora(
23
24              "Lenovo",      // marca
25              "SN-001-XYZ",  // número de serie
26              "B550M",       // modelo de la placa madre
27              "AMD B550"     // chipset
28          );
29
30          // vinculamos r la computadora con el dueño (asociación bidireccional)
31          pc1.setPropietario(propietario1);
32
33          // mostramos datos del propietario usando los getters
34          System.out.println("Propietario");
35          System.out.println("Nombre: " + propietario1.getNombre());
36          System.out.println("DNI: " + propietario1.getDni());
37
38          System.out.println(" ");
39
40          System.out.println("Computador del propietario");
41          System.out.println("Marca: " + propietario1.getComputadora().getMarca());
42          System.out.println("Número de serie: " + propietario1.getComputadora().getNumeroSerie());
43
44          System.out.println(" ");
45
46          System.out.println("\n La Placa madre de la computadora ");
47          System.out.println("Modelo: " + propietario1.getComputadora().getPlacaMadre().getModelo());
48          System.out.println("Chipset: " + propietario1.getComputadora().getPlacaMadre().getChipset());
49
50
51      }
```

Consola

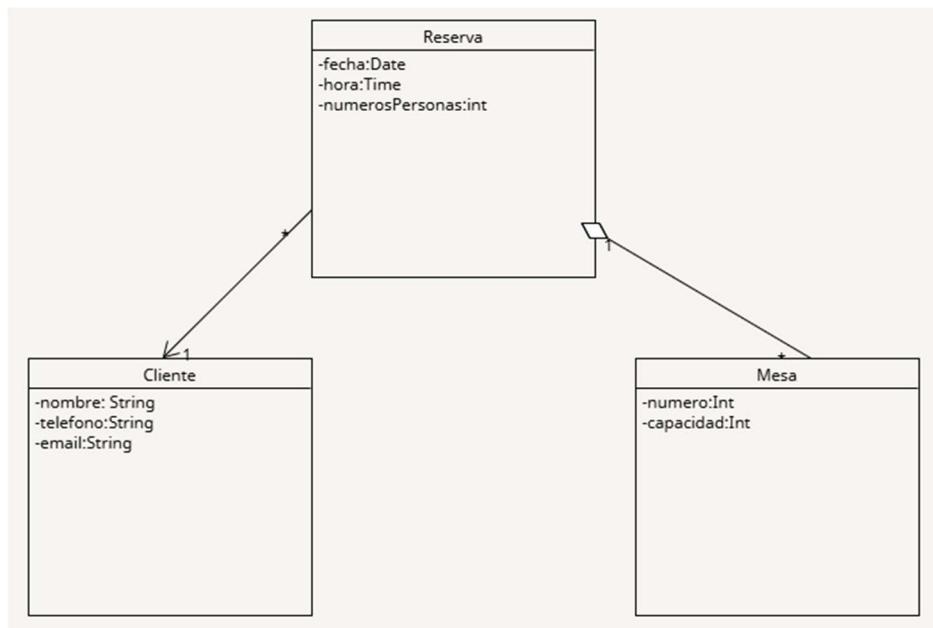
```
Output - 5_Computadora_Placa_Propietario (run) x
▶ run:
▶ Propietario
▶ Nombre: Jhoony Depp
▶ DNI: 12345678

Computador del propietario
Marca: Lenovo
Número de serie: SN-001-XYZ

La Placa madre de la computadora
Modelo: B550M
Chipset: AMD B550
BUILD SUCCESSFUL (total time: 0 seconds)
```

6-Reserva - Cliente - Mesa

- h. Asociación unidireccional: **Reserva → Cliente**
- i. Agregación: **Reserva → Mesa**



Clases y atributos:

i. Reserva: fecha, hora

```
10  /********************************************************************************
11  * @author: [REDACTED]
12  */
13  public class Reserva {
14
15      private String fecha;
16      private String hora;
17
18      // Asociación unidireccional: Reserva -> Cliente
19      private Cliente cliente;
20
21
22      // Constructor
23      public Reserva(String fecha, String hora, Cliente cliente, Mesa mesa) {
24          this.fecha = fecha;
25          this.hora = hora;
26          this.cliente = cliente;
27          this.mesa = mesa;
28      }
29
30      // Getters y setters
31      public String getFecha() {
32          return fecha;
33      }
34
35      public void setFecha(String fecha) {
36          this.fecha = fecha;
37      }
38
39      public String getHora() {
40          return hora;
41      }
42
43      public void setHora(String hora) {
44          this.hora = hora;
45      }
46
47      public Cliente getCliente() {
48          return cliente;
49      }
50
51      public void setCliente(Cliente cliente) {
52          this.cliente = cliente;
53      }
54  }
```

ii. Cliente: nombre, telefono

```
8
9     * @author agust
10    */
11    public class Cliente {
12        private String nombre;
13        private String telefono;
14
15        // Constructor
16        public Cliente(String nombre, String telefono) {
17            this.nombre = nombre;
18            this.telefono = telefono;
19        }
20
21        // Getters y setters
22        public String getNombre() {
23            return nombre;
24        }
25
26        public void setNombre(String nombre) {
27            this.nombre = nombre;
28        }
29
30        public String getTelefono() {
31            return telefono;
32        }
33
34        public void setTelefono(String telefono) {
35            this.telefono = telefono;
36        }
37
38        @Override
39        public String toString() {
40            return "Cliente{" +
41                "nombre='" + nombre + '\'' +
42                ", telefono='" + telefono + '\'' +
43                '}';
44        }
45    }
```

iii. Mesa: numero, capacidad

```
9  * @author agust
10 */
11 public class Mesa {
12     private int numero;
13     private int capacidad;
14
15     // Constructor
16     public Mesa(int numero, int capacidad) {
17         this.numero = numero;
18         this.capacidad = capacidad;
19     }
20
21     // Getters y setters
22     public int getNumero() {
23         return numero;
24     }
25
26     public void setNumero(int numero) {
27         this.numero = numero;
28     }
29
30     public int getCapacidad() {
31         return capacidad;
32     }
33
34     public void setCapacidad(int capacidad) {
35         this.capacidad = capacidad;
36     }
37
38     @Override
39     public String toString() {
40         return "Mesa{" +
41             "numero=" + numero +
42             ", capacidad=" + capacidad +
43             '}';
44     }
45 }
46 }
```

Main

```

13  /*
14  * @param args the command line arguments
15  */
16 public static void main(String[] args) {
17     // Crear un cliente
18     Cliente cliente1 = new Cliente("Pepito Aviles", "011-151234567");
19
20     // Crear una mesa (agregación: existe fuera de la reserva)
21     Mesa mesa1 = new Mesa(5, 4);
22
23     // Crear una reserva asociando cliente y mesa
24     Reserva reserva1 = new Reserva(
25         "2025-11-20",
26         "21:30",
27         cliente1,
28         mesa1
29     );
30
31     // Mostrar la información
32     System.out.println("reserva ");
33     System.out.println("Fecha: " + reserva1.getFecha());
34     System.out.println("Hora: " + reserva1.getHora());
35
36     System.out.println("\nCliente ");
37     System.out.println("Nombre: " + reserva1.getCliente().getNombre());
38     System.out.println("Teléfono: " + reserva1.getCliente().getTelefono());
39
40     System.out.println(" ");
41
42
43     System.out.println("Mesa ");
44     System.out.println("Número: " + reserva1.getMesa().getNumero());
45     System.out.println("Capacidad: " + reserva1.getMesa().getCapacidad());
46
47     System.out.println("\n--- toString completo ---");
48     System.out.println(reserva1);
49 }
50 }
```

Print por consola

```

Output - 6_Reserva_Cliente_mesa (run) ×
▶ reserva
▶ Fecha: 2025-11-20
▶ Hora: 21:30
✖
Cliente
Nombre: Pepito Aviles
Teléfono: 011-151234567

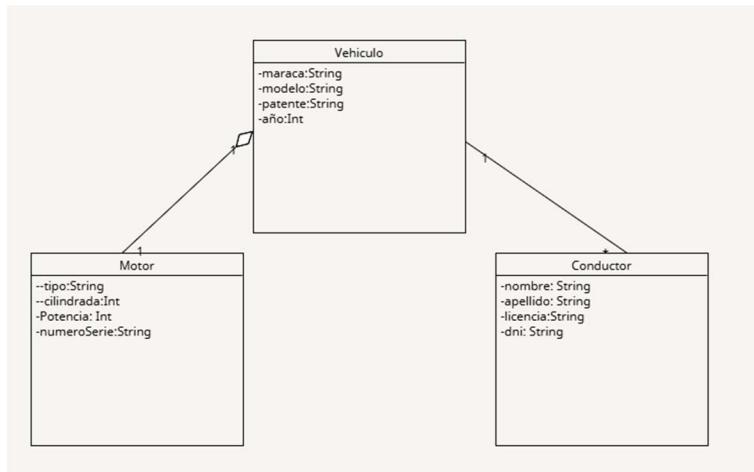
Mesa
Número: 5
Capacidad: 4

--- toString completo ---
Reserva{fecha='2025-11-20', hora='21:30', cliente=Cliente{nombre='Pepito Aviles', telefono='011-151234567'}}

BUILD SUCCESSFUL (total time: 0 seconds)
```

7-Vehículo - Motor - Conductor

- j. Agregación: **Vehículo → Motor**
- k. Asociación bidireccional: **Vehículo ↔ Conductor**



Clases y atributos:

- i. Vehículo: patente, modelo

```

11 public class Vehiculo {
12     // Atributos básicos
13     private String patente;
14     private String modelo;
15
16     // Agregación: un Vehiculo tiene un Motor, pero el Motor puede existir aparte
17     private Motor motor;
18
19     // Asociación bidireccional con Conductor
20     private Conductor conductor;
21
22     // Constructor de vehiculo
23     public Vehiculo(String patente, String modelo, Motor motor) {
24         this.patente = patente;
25         this.modelo = modelo;
26         this.motor = motor;
27     }
28
29     // Getters y setters
30     public String getPatente() {
31         return patente;
  
```

```
32     }
33
34     public void setPatente(String patente) {
35         this.patente = patente;
36     }
37
38     public String getModelo() {
39         return modelo;
40     }
41
42     public void setModelo(String modelo) {
43         this.modelo = modelo;
44     }
45
46     public Motor getMotor() {
47         return motor;
48     }
49
50     public void setMotor(Motor motor) {
51         this.motor = motor;
52     }
53
54 // Asociación bidireccional: Vehiculo <-> Conductor
55     public Conductor getConductor() {
56         return conductor;
57     }
58
59     public void setConductor(Conductor conductor) {
60         this.conductor = conductor;
61         // Si el conductor no tiene seteado este vehículo, lo sincronizamos
62         if (conductor != null && conductor.getVehiculo() != this) {
63             conductor.setVehiculo(this);
64         }
65     }
66
67 // Asociación bidireccional: Vehiculo <-> Conductor
68     public Conductor getConductor() {
69         return conductor;
70     }
71
72     public void setConductor(Conductor conductor) {
73         this.conductor = conductor;
74         // Si el conductor no tiene seteado este vehículo, lo sincronizamos
75         if (conductor != null && conductor.getVehiculo() != this) {
76             conductor.setVehiculo(this);
77         }
78     }
79 
```

ii. Motor: tipo, numeroSerie

```
10
11     public class Motor {
12         private String tipo;
13         private String numeroSerie;
14
15         // Constructor del motor
16         public Motor(String tipo, String numeroSerie) {
17             this.tipo = tipo;
18             this.numeroSerie = numeroSerie;
19         }
20
21         // Getters y setters
22         public String getTipo() {
23             return tipo;
24         }
25
26         public void setTipo(String tipo) {
27             this.tipo = tipo;
28         }
29
30         public String getNumeroSerie() {
31             return numeroSerie;
32         }
33
34         public void setNumeroSerie(String numeroSerie) {
35             this.numeroSerie = numeroSerie;
36         }
37
38         @Override
39         public String toString() {
40             return "Motor{" +
41                 "tipo='" + tipo + '\'' +
42                 ", numeroSerie='" + numeroSerie + '\'' +
43                 '}';
44         }
45     }
46 }
```

iii. Conductor: nombre, licencia

```
10 | */  
11 | public class Conductor {  
12 |     private String nombre;  
13 |     private String licencia;  
14 |  
15 |     // Aca tenemos una Asociación bidireccional: Conductor <-> Vehiculo  
16 |     private Vehiculo vehiculo;  
17 |  
18 |     // Constructor de la clase|  
19 |     public Conductor(String nombre, String licencia) {  
20 |         this.nombre = nombre;  
21 |         this.licencia = licencia;  
22 |     }  
23 |  
24 |     // Getters y setters  
25 |     public String getNombre() {  
26 |         return nombre;  
27 |     }  
28 |  
29 |     public void setNombre(String nombre) {  
30 |         this.nombre = nombre;  
31 |     }  
32 |  
33 |     public String getLicencia() {  
34 |         return licencia;  
35 |     }  
36 |  
37 |     public void setLicencia(String licencia) {  
38 |         this.licencia = licencia;  
39 |     }  
40 |  
41 |     public Vehiculo getVehiculo() {  
42 |         return vehiculo;  
43 |     }  
44 |
```



```

45  public void setVehiculo(Vehiculo vehiculo) {
46      this.vehiculo = vehiculo;
47      // Sincronizamos el otro lado de la relación
48      if (vehiculo != null && vehiculo.getConductor() != this) {
49          vehiculo.setConductor(this);
50      }
51  }
52  //to string
53  @Override
54  public String toString() {
55      return "Conductor{" +
56          "nombre='" + nombre + '\'' +
57          ", licencia='" + licencia + '\'' +
58          '}';
59  }
60 }
61

```

Main

```

15
16  public static void main(String[] args) {
17      // Creamos un motor (agregación: existe fuera del vehículo)
18      Motor motor1 = new Motor("Nafta 1.6", "MTR-12345");
19
20      // instanciamos un vehículo y asociarle el motor
21      Vehiculo vehiculo1 = new Vehiculo("ABC123", "Toyota Corolla", motor1);
22
23      // creamos conductor
24      Conductor conductor1 = new Conductor("Moria Casan", "LIC-124");
25
26      // Asociamos vehículo y conductor (bidireccional)
27      vehiculo1.setConductor(conductor1);
28
29
30      System.out.println("-----Vehículo -----");
31      System.out.println("Patente: " + vehiculo1.getPatente());
32      System.out.println("Modelo: " + vehiculo1.getModelo());
33
34      System.out.println("\n+++++ Motor ++++");
35      System.out.println("Tipo: " + vehiculo1.getMotor().getTipo());
36      System.out.println("Número de serie: " + vehiculo1.getMotor().getNumeroSerie());
37
38      System.out.println("\n*****Conductor *****");
39      System.out.println("Nombre: " + vehiculo1.getConductor().getNombre());
40      System.out.println("Licencia: " + vehiculo1.getConductor().getLicencia());
41
42      System.out.println("\n--- toString() ---");
43      System.out.println(vehiculo1);
44      System.out.println(conductor1);
45  }
46 }

```

Print por consola

```
Output - Vehiculo_Motor_Conductor (run) x
run:
-----Vehículo -----
Patente: ABC123
Modelo: Toyota Corolla

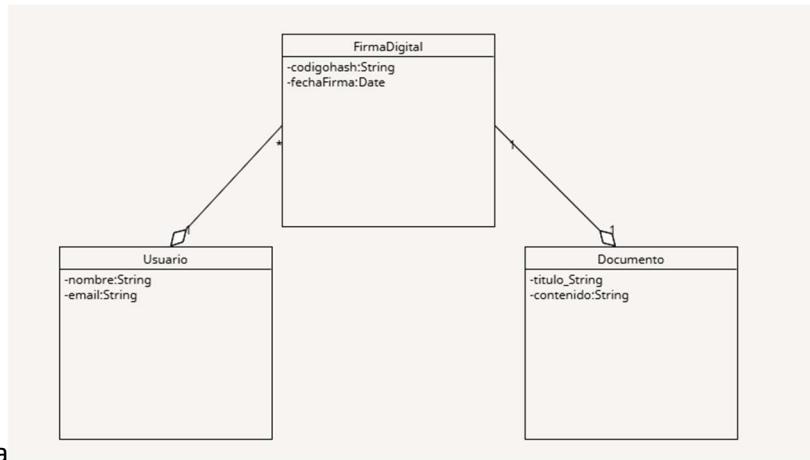
++++ Motor +++
Tipo: Nafta 1.6
Número de serie: MTR-12345

*****Conductor *****
Nombre: Moria Casan
Licencia: LIC-124

--- toString() ---
Vehiculo{patente='ABC123', modelo='Toyota Corolla', motor=Motor{tipo='Nafta 1.6', numeroSerie='MTR-12345', conductor=Conductor{nombre='Moria Casan', licencia='LIC-124'}}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

8-Documento - FirmaDigital - Usuario

- i. Composición: **Documento → FirmaDigital**
- m. Agregación: **FirmaDigital → Usuario**



n. Firma

Clases y atributos:

- i. Documento: titulo, contenido

```
10  */
11 public class Documento {
12     private String titulo;
13     private String contenido;
14
15     // Composición: el Documento contiene una FirmaDigital
16     private FirmaDigital firmaDigital;
17
18     // Constructor SIN firma todavía
19     public Documento(String titulo, String contenido) {
20         this.titulo = titulo;
21         this.contenido = contenido;
22     }
23
24     // Getters y setters
25     public String getTitulo() {
26         return titulo;
27     }
28
29     public void setTitulo(String titulo) {
30         this.titulo = titulo;
31     }
32
33     public String getContenido() {
34         return contenido;
35     }
36
37     public void setContenido(String contenido) {
38         this.contenido = contenido;
39     }
40
41     public FirmaDigital getFirmaDigital() {
42         return firmaDigital;
43     }
44
45     // Método para "firmar" el documento (crea la firma: COMPOSICIÓN)
46     public void firmarDocumento(String codigoHash, String fecha, Usuario usuario) {
47         this.firmaDigital = new FirmaDigital(codigoHash, fecha, usuario);
48     }
49
50     @Override
51     public String toString() {
52         return "Documento{" +
53             "titulo='" + titulo + '\'' +
54             ", contenido='" + contenido + '\'' +
55             ", firmaDigital=" + firmaDigital +
56             '}';
57     }
58 }
```

ii. FirmaDigital: codigoHash, fecha

```
9      * @author agust
10     */
11    public class FirmaDigital {
12      private String codigoHash;
13      private String fecha;
14
15      // Agregación: la FirmaDigital se asocia a un Usuario existente
16      private Usuario usuario;
17
18      public FirmaDigital(String codigoHash, String fecha, Usuario usuario) {
19          this.codigoHash = codigoHash;
20          this.fecha = fecha;
21          this.usuario = usuario;
22      }
23
24      // Getters y setters
25      public String getCodigoHash() {
26          return codigoHash;
27      }
28
29      public void setCodigoHash(String codigoHash) {
30          this.codigoHash = codigoHash;
31      }
32
33      public String getFecha() {
34          return fecha;
35      }
36
37      public void setFecha(String fecha) {
38          this.fecha = fecha;
39      }
40
41      public Usuario getUsuario() {
42          return usuario;
43      }
44
45      public void setUsuario(Usuario usuario) {
46          this.usuario = usuario;
47      }
48
49      @Override
50      public String toString() {
51          return "FirmaDigital{" +
52              "codigoHash='" + codigoHash + '\'' +
53              ", fecha='" + fecha + '\'' +
54              ", usuario='" + usuario +
55              '}';
56      }
57
58  }
```

iii. Usuario: nombre, email

```
10
11     public class Usuario {
12         private String nombre;
13         private String email;
14
15         public Usuario(String nombre, String email) {
16             this.nombre = nombre;
17             this.email = email;
18         }
19
20         // Getters y setters
21         public String getNombre() {
22             return nombre;
23         }
24
25         public void setNombre(String nombre) {
26             this.nombre = nombre;
27         }
28
29         public String getEmail() {
30             return email;
31         }
32
33         public void setEmail(String email) {
34             this.email = email;
35         }
36
37         @Override
38         public String toString() {
39             return "Usuario{" +
40                 "nombre='" + nombre + '\'' +
41                 ", email='" + email + '\'' +
42                 '}';
43         }
44     }
```

Main

```

15  /
16  public static void main(String[] args) {
17      // creamos usuario
18      Usuario usuario1 = new Usuario("Ana López", "ana_lopez2025@gmail.com");
19
20      // Creamos un documento (todavía sin firma)
21      Documento doc1 = new Documento(
22          "Contrato de Servicios",
23          "Contenido del contrato..."
24      );
25
26      // Firm el documento (composición: el doc crea la FirmaDigital)
27      doc1.firmarDocumento(
28          "ABC123HASH", // código hash simulado
29          "2025-11-20", // fecha de firma
30          usuario1 // usuario que firma (agregación)
31      );
32
33      // linea de print
34      System.out.println("==> Documento ==>");
35      System.out.println("Título: " + doc1.getTitulo());
36      System.out.println("Contenido: " + doc1.getContenido());
37
38      System.out.println("\n==> Firma Digital ==>");
39      System.out.println("Hash: " + doc1.getFirmaDigital().getCodigoHash());
40      System.out.println("Fecha: " + doc1.getFirmaDigital().getFecha());
41
42      System.out.println("\n==> Usuario que firma ==>");
43      System.out.println("Nombre: " + doc1.getFirmaDigital().getUsuario().getNombre());
44      System.out.println("Email: " + doc1.getFirmaDigital().getUsuario().getEmail());
45
46      System.out.println("\n--- toString completo ---");
47      System.out.println(doc1);
48  }
49
}

```

Print Consola

```

Output - Documento_FirmaDigital_Usuario (run) ×
▶ run:
▶ ==> Documento ==>
[T]ítulo: Contrato de Servicios
[+] Contenido: Contenido del contrato...
|
==> Firma Digital ==
Hash: ABC123HASH
Fecha: 2025-11-20

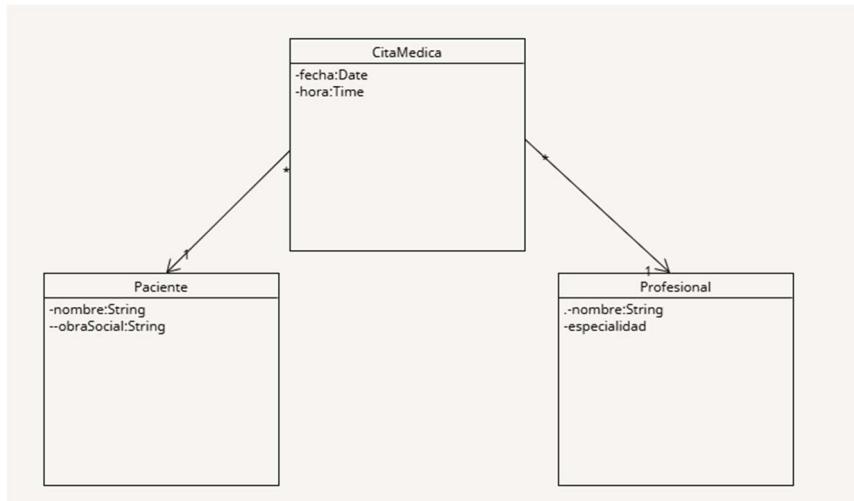
==> Usuario que firma ==
Nombre: Ana López
Email: ana_lopez2025@gmail.com

--- toString completo ---
Documento{titulo='Contrato de Servicios', contenido='Contenido del contrato...', firmaDigital=FirmaDigital@4d5...}
BUILD SUCCESSFUL (total time: 0 seconds)

```

9-CitaMédica - Paciente - Profesional

- o. Asociación unidireccional: **CitaMédica → Paciente**,
- p. Asociación unidireccional: **CitaMédica → Profesional**



Clases y atributos:

- i. CitaMédica: fecha, hora

```

11 public class CitaMedica {
12
13     // Atributos propios de la cita
14     private String fecha; // por simplicidad, usamos String
15     private String hora;
16
17     // Asociación unidireccional: la cita conoce al paciente
18     private Paciente paciente;
19
20     // Asociación unidireccional: la cita conoce al profesional
21     private Profesional profesional;
22
23     // Constructor: se crea una cita con fecha, hora, paciente y profesional
24     public CitaMedica(String fecha, String hora, Paciente paciente, Profesional profesional) {
25         this.fecha = fecha;
26         this.hora = hora;
27         this.paciente = paciente;
28         this.profesional = profesional;
29     }
30
  
```

```
31 // Getters y setters de fecha y hora
32     public String getFecha() {
33         return fecha;
34     }
35
36     public void setFecha(String fecha) {
37         this.fecha = fecha;
38     }
39
40     public String getHora() {
41         return hora;
42     }
43
44     public void setHora(String hora) {
45         this.hora = hora;
46     }
47
48 // Getters y setters de paciente y profesional
49     public Paciente getPaciente() {
50         return paciente;
51     }
52
53     public void setPaciente(Paciente paciente) {
54         this.paciente = paciente;
55     }
56
57     public Profesional getProfesional() {
58         return profesional;
59     }
60
61     public void setProfesional(Profesional profesional) {
62         this.profesional = profesional;
63     }
64
65     @Override
66     public String toString() {
67         return "CitaMedica{" +
68             "fecha=" + fecha + '\n' +
69             ", hora=" + hora + '\n' +
70             ", paciente=" + paciente +
71             ", profesional=" + profesional +
72             '}';
73     }
74 }
```

ii. Paciente: nombre, obraSocial

```
11 public class Paciente {  
12     // Atributos del paciente  
13     private String nombre;  
14     private String obraSocial;  
15  
16     // Constructor: se crea un paciente con nombre y obra social  
17     public Paciente(String nombre, String obraSocial) {  
18         this.nombre = nombre;  
19         this.obraSocial = obraSocial;  
20     }  
21  
22     // Getter y setter de nombre  
23     public String getNombre() {  
24         return nombre;  
25     }  
26  
27     public void setNombre(String nombre) {  
28         this.nombre = nombre;  
29     }  
30  
31     // Getter y setter de obraSocial  
32     public String getObraSocial() {  
33         return obraSocial;  
34     }  
35  
36     public void setObraSocial(String obraSocial) {  
37         this.obraSocial = obraSocial;  
38     }  
39  
40     @Override  
41     public String toString() {  
42         return "Paciente{" +  
43             "nombre='" + nombre + '\'' +  
44             ", obraSocial='" + obraSocial + '\'' +  
45             '}';  
46     }  
47 }
```

iii. Profesional: nombre, especialidad

```
11 public class Profesional {  
12     // Atributos del profesional  
13     private String nombre;  
14     private String especialidad;  
15  
16     // Constructor: se crea un profesional con nombre y especialidad  
17     public Profesional(String nombre, String especialidad) {  
18         this.nombre = nombre;  
19         this.especialidad = especialidad;  
20     }  
21  
22     // Getter y setter de nombre  
23     public String getNombre() {  
24         return nombre;  
25     }  
26  
27     public void setNombre(String nombre) {  
28         this.nombre = nombre;  
29     }  
30 }
```

```
31     // Getter y setter de especialidad  
32     public String getEspecialidad() {  
33         return especialidad;  
34     }  
35  
36     public void setEspecialidad(String especialidad) {  
37         this.especialidad = especialidad;  
38     }  
39  
40     @Override  
41     public String toString() {  
42         return "Profesional{" +  
43             "nombre=" + nombre + '\'' +  
44             ", especialidad=" + especialidad + '\'' +  
45             '}';  
46     }  
47 }
```

mAIN

```

15    /*
16     * public static void main(String[] args) {
17     *
18     * // 1) Creamos un paciente
19     * Paciente paciente1 = new Paciente("Osvaldo Laport", "OSDE 310");
20     *
21     * // 2) Creamos un profesional
22     * Profesional profesional1 = new Profesional("Dra. López", "Clínica Médica");
23     *
24     * // 3) Creamos una cita médica que une paciente + profesional
25     * CitaMedica cita1 = new CitaMedica(
26     *         "2025-11-25", // fecha de la cita
27     *         "10:30", // hora de la cita
28     *         paciente1, // paciente asociado
29     *         profesional1 // profesional asociado
30     * );
31
32     * // 4) Mostramos los datos de la cita
33     * System.out.println("==> Cita médica ==>");
34     * System.out.println("Fecha: " + cita1.getFecha());
35     * System.out.println("Hora: " + cita1.getHora());
36
37     * // 5) Mostramos los datos del paciente de esa cita
38     * System.out.println("\n==> Paciente ==>");
39     * System.out.println("Nombre: " + cita1.getpaciente().getNombre());
40     * System.out.println("Obra social: " + cita1.getpaciente().getObraSocial());
41
42     * // 6) Mostramos los datos del profesional de esa cita
43     * System.out.println("\n==> Profesional ==>");
44     * System.out.println("Nombre: " + cita1.getProfesional().getNombre());
45     * System.out.println("Especialidad: " + cita1.getProfesional().getEspecialidad());
46
47     * // 7) Mostramos todo junto usando toString()
48     * System.out.println("\n--- toString completo ---");
49     * System.out.println(cita1);
50   }
51 }
```

Print consola

```

Output - 9_CitaMedica_Paciente_Profesional (run) x
▶ run:
▶ === Cita médica ===
▶ Fecha: 2025-11-25
▶ Hora: 10:30

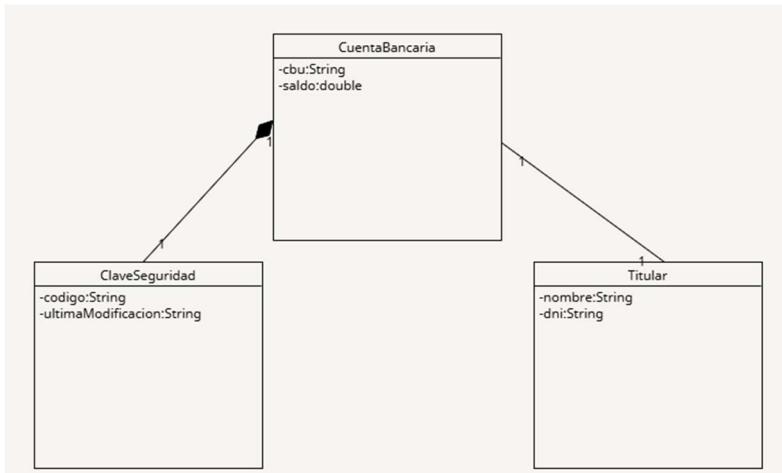
==== Paciente ====
Nombre: Osvaldo Laport
Obra social: OSDE 310

==== Profesional ====
Nombre: Dra. López
Especialidad: Clínica Médica

--- toString completo ---
CitaMedica{fecha='2025-11-25', hora='10:30', paciente=Paciente{nombre='Osvaldo Laport', obraSocial='OS
BUILD SUCCESSFUL (total time: 0 seconds)
```

10-CuentaBancaria - ClaveSeguridad - Titular

- q. Composición: **CuentaBancaria → ClaveSeguridad**
- r. Asociación bidireccional: **CuentaBancaria ↔ Titular**



Clases y atributos:

- i. CuentaBancaria: cbu, saldo

```

10
11 public class CuentaBancaria {
12     // Atributos propios de la cuenta
13     private String cbu;
14     private double saldo;
15
16     // Composición: la Cuenta "contiene" una ClaveSeguridad
17     private ClaveSeguridad claveSeguridad;
18
19     // Asociación bidireccional: CuentaBancaria <-> Titular
20     private Titular titular;
21
22     // Constructor: creamos la cuenta y su clave de seguridad ,composición
23     public CuentaBancaria(String cbu, double saldo, String codigoClave, String ultimaModificacion) {
24         this.cbu = cbu;
25         this.saldo = saldo;
26         // La cuenta crea su propia clave - composición
27         this.claveSeguridad = new ClaveSeguridad(codigoClave, ultimaModificacion);
28     }
29
  
```

```
30 // Getters y setters básicos
31     public String getCbu() {
32         return cbu;
33     }
34
35     public void setCbu(String cbu) {
36         this.cbu = cbu;
37     }
38
39     public double getSaldo() {
40         return saldo;
41     }
42
43     public void setSaldo(double saldo) {
44         this.saldo = saldo;
45     }
46
47     public ClaveSeguridad getClaveSeguridad() {
48         return claveSeguridad;
49     }
50
51 // Método para actualizar la clave (sigue siendo composición)
52     public void actualizarClave(String nuevoCodigo, String nuevaFecha) {
53         this.claveSeguridad.setCodigo(nuevoCodigo);
54         this.claveSeguridad.setUltimaModificacion(nuevaFecha);
55     }
56
57 // Asociación bidireccional con Titular
58     public Titular getTitular() {
59         return titular;
60     }
61
62     public void setTitular(Titular titular) {
63         this.titular = titular;
64         // relacionamos con la tabla titular
65         if (titular != null && titular.getCUENTA_BANCARIA() != this) {
66             titular.setCUENTA_BANCARIA(this);
67         }
68     }
69
70     @Override
71     public String toString() {
72         return "CuentaBancaria{" +
73             "cbu='" + cbu + '\'' +
74             ", saldo=" + saldo +
75             ", claveSeguridad=" + claveSeguridad +
76             ", titular=" + (titular != null ? titular.getNombre() : "sin titular") +
77             '}';
78     }
79 }
```

ii. ClaveSeguridad: codigo, ultimaModificacion

```
11 public class ClaveSeguridad {  
12  
13     // Representamos la clave de seguridad asociada a una cuenta  
14     private String codigo;  
15     private String ultimaModificacion;  
16  
17  
18     // Constructor  
19     public ClaveSeguridad(String codigo, String ultimaModificacion) {  
20         this.codigo = codigo;  
21         this.ultimaModificacion = ultimaModificacion;  
22     }  
23  
24     // Getters y setters  
25     public String getCodigo() {  
26         return codigo;  
27     }  
28  
29  
30  
31     public void setCodigo(String codigo) {  
32         this.codigo = codigo;  
33     }  
34  
35     public String getUltimaModificacion() {  
36         return ultimaModificacion;  
37     }  
38  
39     public void setUltimaModificacion(String ultimaModificacion) {  
40         this.ultimaModificacion = ultimaModificacion;  
41     }  
42  
43     @Override  
44     public String toString() {  
45         return "ClaveSeguridad{" +  
46             "codigo=" + codigo + '\'' +  
47             ", ultimaModificacion=" + ultimaModificacion + '\'' +  
48             '}';  
49     }  
50 }
```



iii. iii. Titular: nombre, dni.

```
11 public class Titular {  
12     private String nombre;  
13     private String dni;  
14  
15     // Asociación bidireccional: Titular <-> CuentaBancaria  
16     private CuentaBancaria cuentaBancaria;  
17  
18     // Constructor  
19     public Titular(String nombre, String dni) {  
20         this.nombre = nombre;  
21         this.dni = dni;  
22     }  
23  
24     // Getters y setters básicos  
25     public String getNombre() {  
26         return nombre;  
27     }  
28  
29     public void setNombre(String nombre) {  
30         this.nombre = nombre;  
31     }  
32 }
```

```
32  
33     public String getDni() {  
34         return dni;  
35     }  
36  
37     public void setDni(String dni) {  
38         this.dni = dni;  
39     }  
40  
41     public CuentaBancaria getCuentaBancaria() {  
42         return cuentaBancaria;  
43     }  
44  
45     public void setCuentaBancaria(CuentaBancaria cuentaBancaria) {  
46         this.cuentaBancaria = cuentaBancaria;  
47         // Sincronizamos el otro lado si hace falta  
48         if (cuentaBancaria != null && cuentaBancaria.getTitular() != this) {  
49             cuentaBancaria.setTitular(this);  
50         }  
51     }
```



```

52
53     @Override
54     public String toString() {
55         return "Titular{" +
56             "nombre=" + nombre + '\n' +
57             ", dni=" + dni + '\n' +
58             '}';
59     }
60 }
```

Main

```

16     public static void main(String[] args) {
17         // 1) Creamos un titular
18         Titular titular1 = new Titular("Ricardo Fort", "30123456");
19
20         // 2) Creamos una cuenta bancaria con su clave de seguridad (composición)
21         CuentaBancaria cuenta1 = new CuentaBancaria(
22             "0001234500001234567890", // cbu
23             800000.50,               // saldo
24             "ABC123",                // código de la clave
25             "2025-11-20"              // fecha de última modificación
26         );
27
28         // 3) Asociamos cuenta y titular (bidireccional)
29         cuenta1.setTitular(titular1);
30
31         // 4) Mostramos datos básicos de la cuenta
32         System.out.println("==> Cuenta bancaria ==>");
33         System.out.println("CBU: " + cuenta1.getCBU());
34         System.out.println("Saldo: " + cuenta1.getSaldo());
35
36         // 5) Mostramos datos de la clave de seguridad
37         System.out.println("\n==> Clave de seguridad ==>");
38         System.out.println("Código: " + cuenta1.getClaveSeguridad().getCodigo());
39         System.out.println("Última modificación: " + cuenta1.getClaveSeguridad().getUltimaModificacion());
40
41         // 6) Mostramos datos del titular
42         System.out.println("\n==> Titular ==>");
43         System.out.println("Nombre: " + cuenta1.getTitular().getNombre());
44         System.out.println("DNI: " + cuenta1.getTitular().getDNI());
45
46
47         // actualizacimos la clave
48         cuenta1.actualizarClave("XYZ789", "2025-12-01");
49         System.out.println("\nClave actualizada: " + cuenta1.getClaveSeguridad());
50
51     }
52 }
```

Print consola

```
Output - 10_CuentaBancaria_ClaveSeguridad_Titular (run) x
▶ run:
▶ === Cuenta bancaria ===
▶ CBU: 0001234500001234567890
▶ Saldo: 800000.5

    === Clave de seguridad ===
    Código: ABC123
    Última modificación: 2025-11-20

    === Titular ===
    Nombre: Ricardo Fort
    DNI: 30123456

    Clave actualizada: ClaveSeguridad{codigo='XYZ789', ultimaModificacion='2025-12-01'}
BUILD SUCCESSFUL (total time: 0 seconds)
```

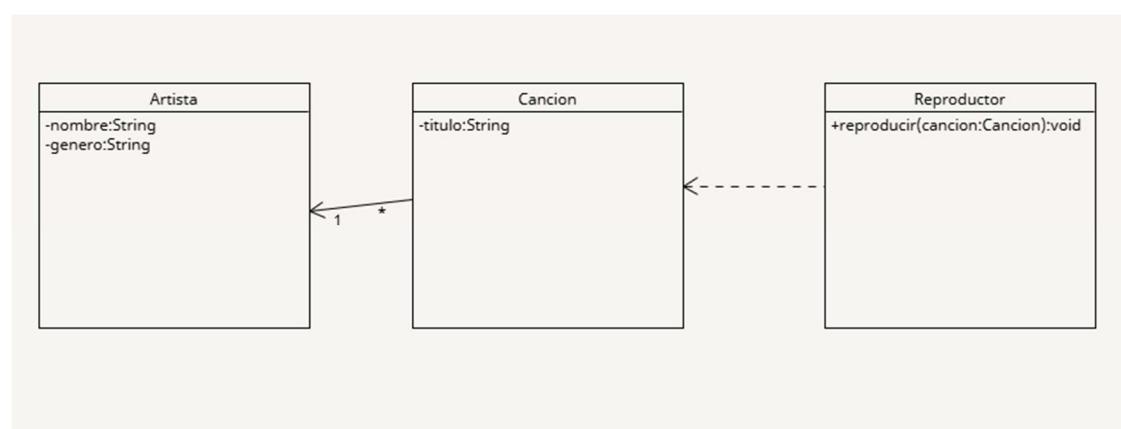
DEPENDENCIA DE USO

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

Ejercicios de Dependencia de Uso

11. Reproductor - Canción - Artista

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



c.

Clases y atributos:

i. Canción: titulo.

```
10  /
11  public class Cancion {
12      private String titulo;
13
14      // Asociación unidireccional: Canción -> Artista
15      private Artista artista;
16
17      // Constructor de la clases
18      public Cancion(String titulo, Artista artista) {
19          this.titulo = titulo;
20          this.artista = artista;
21      }
22
23      // Getters y setters
24      public String getTitulo() {
25          return titulo;
26      }
27
28      public void setTitulo(String titulo) {
29          this.titulo = titulo;
30      }
31
32      public Artista getArtista() {
33          return artista;
34      }
35
36      public void setArtista(Artista artista) {
37          this.artista = artista;
38      }
39
40      @Override
41      public String toString() {
42          return "Cancion{" +
43              "titulo='" + titulo + '\'' +
44              ", artista=" + artista +
45              '}';
46      }
47  }
```

ii. Artista: nombre, genero.

```

11 public class Artista {
12     private String nombre;
13     private String genero;
14
15     // Constructor de la clase
16     public Artista(String nombre, String genero) {
17         this.nombre = nombre;
18         this.genero = genero;
19     }
20
21     // Getters y setters
22     public String getNombre() {
23         return nombre;
24     }
25
26     public void setNombre(String nombre) {
27         this.nombre = nombre;
28     }
29
30     public String getGenero() {
31         return genero;
32     }
33
34     public void setGenero(String genero) {
35         this.genero = genero;
36     }
37
38     @Override
39     public String toString() {
40         return "Artista{" +
41             "nombre='" + nombre + '\'' +
42             ", genero='" + genero + '\'' +
43             '}';
44     }
45 }
```

iii. Reproductor->método: void reproducir(Cancion cancion)

```

8  *
9  * @author agust
10 */
11 public class Reproductor {
12     // Dependencia de uso: nuestro método recibe una Cancion para reproducirla
13     public void reproducir(Cancion cancion) {
14         System.out.println("Reproduciendo: " + cancion.getTitulo());
15         System.out.println("Artista: " + cancion.getArtista().getNombre());
16     }
17 }
```



Main

```

13  /*
14   * @param args the command line arguments
15   */
16  public static void main(String[] args) {
17      // Creamos un artista
18      Artista artista1 = new Artista("Patricio Rey", "Rock");
19
20      // creamos una canción asociada a ese artista
21      Cancion cancion1 = new Cancion("MotorPsico", artista1);
22
23      // creamos un reproductor
24      Reproductor reproductor = new Reproductor();
25
26      // por ultimo reproducimos la canción (dependencia de uso)
27      reproductor.reproducir(cancion1);
28  }
29 }
30
31

```

Print consola

Output - 11_Reproductor_Cancion_Artista (run) ×

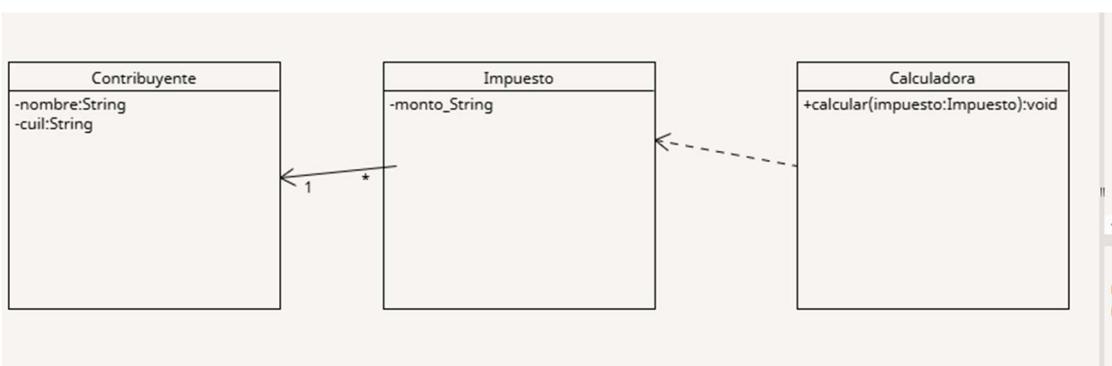
```

▶ run:
▶ Reproduciendo: MotorPsico
▶ Artista: Patricio Rey
▶ BUILD SUCCESSFUL (total time: 0 seconds)

```

12. Impuesto - Contribuyente - Calculadora

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



C.

Clases y atributos:

i. Impuesto: monto.

```
10  />
11  public class Impuesto {
12      private double monto;
13      private Contribuyente contribuyente;
14
15      public Impuesto(double monto, Contribuyente contribuyente) {
16          this.monto = monto;
17          this.contribuyente = contribuyente;
18      }
19
20      public Impuesto() {
21      }
22
23      public double getMonto() {
24          return monto;
25      }
26
27      public void setMonto(double monto) {
28          this.monto = monto;
29      }
30
31      public Contribuyente getContribuyente() {
32          return contribuyente;
33      }
34
35      public void setContribuyente(Contribuyente contribuyente) {
36          this.contribuyente = contribuyente;
37      }
38
39
40
41  }
```

ii. Contribuyente: nombre, cuil.

```
11 public class Contribuyente {  
12  
13     private String cuil;  
14     private String nombre;  
15     // Constructor: creamos un contribuyente con nombre y CUIL  
16     public Contribuyente(String nombre, String cuil) {  
17         this.nombre = nombre;  
18         this.cuil = cuil;  
19     }  
20  
21     // Getters y setters  
22     public String getNombre() {  
23         return nombre;  
24     }  
25  
26     public void setNombre(String nombre) {  
27         this.nombre = nombre;  
28     }  
29  
30     public String getCuil() {  
31         return cuil;  
32     }
```

```
33     public void setCuil(String cuil) {  
34         this.cuil = cuil;  
35     }  
36 }  
37  
38  
39 }  
40 }
```

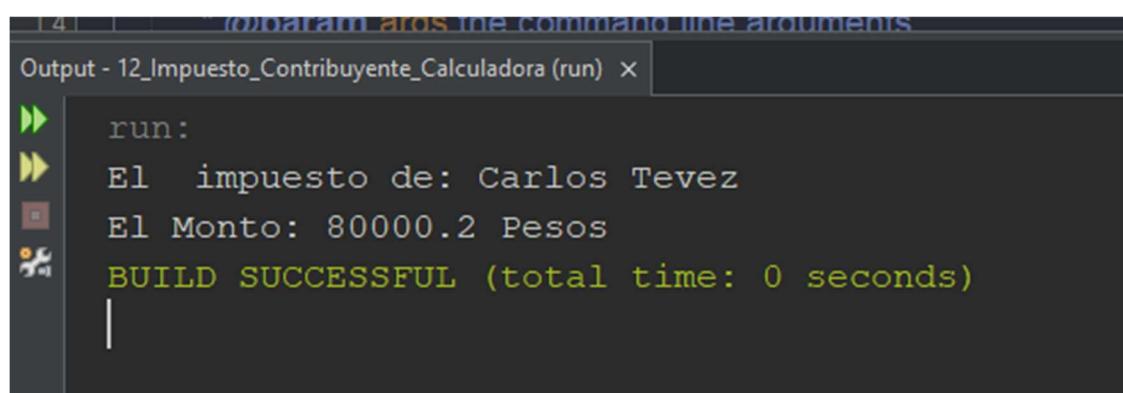
Imp Calculadora->método: void calcular(Impuesto impuesto)

```
/*  
 * @author agust  
 */  
public class Calculadora {  
    public void calcular(Impuesto impuesto) {  
        System.out.println("El impuesto de: " + impuesto.getContribuyente().getNombre());  
        System.out.println("El Monto: " + impuesto.getMonto()+" Pesos");  
    }  
}
```

Main

```
1  public class Main {  
2  
3      /**  
4      * @param args the command line arguments  
5      */  
6      public static void main(String[] args) {  
7          //Creamos un contribuyente  
8          Contribuyente contribuyente = new Contribuyente("Carlos Tevez", "20-56656787-9");  
9          //Creamos un im puesto  
10         Impuesto impuesto = new Impuesto(80000.2, contribuyente);  
11         //Creamos calculadora  
12         Calculadora calc = new Calculadora();  
13         calc.calcular(impuesto); // dependencia de uso  
14     }  
15 }  
16 }
```

Print Consola



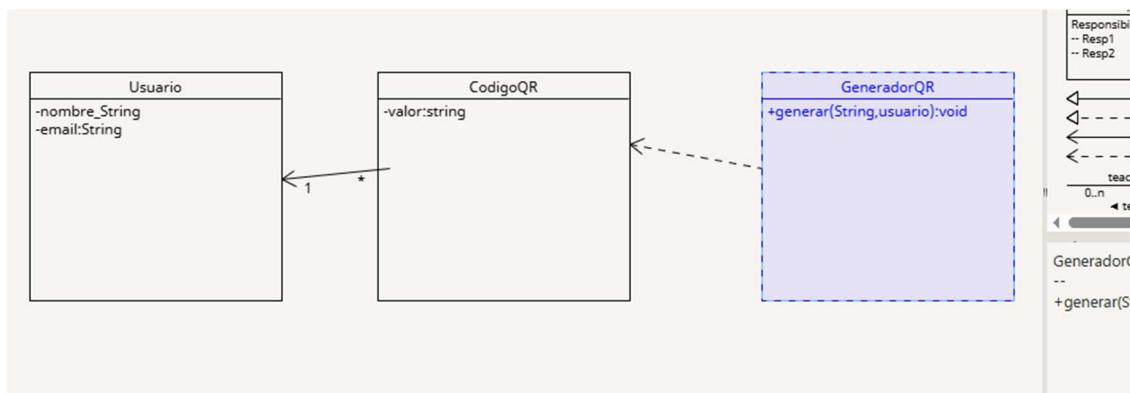
```
Output - 12_Impuesto_Contribuyente_Calculadora (run) ×  
run:  
El impuesto de: Carlos Tevez  
El Monto: 80000.2 Pesos  
BUILD SUCCESSFUL (total time: 0 seconds)
```

DEPENDENCIA DE CREACIÓN

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

Ejercicios de Dependencia de Creación

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



c.

Clases y atributos:

i. CódigoQR: valor.

```

12
13 // Atributos
14 private String valor; // texto codificado en el QR
15 private Usuario usuario; // asociación unidireccional hacia Usuario
16
17 // Constructor
18 public CódigoQR(String valor, Usuario usuario) {
19     this.valor = valor;
20     this.usuario = usuario;
21 }
22
23 // Getters y setters
24 public String getValor() {
25     return valor;
26 }
27
28 public void setValor(String valor) {
29     this.valor = valor;
30 }
31
32 public Usuario getUsuario() {
33     return usuario;
34 }
  
```

```
35
36     public void setUsuario(Usuario usuario) {
37         this.usuario = usuario;
38     }
39     // Método para mostrar información del código QR
40     @Override
41     public String toString() {
42         return "CodigoQR{valor=" + valor + ", usuario=" + usuario + "}";
43     }
44 }
45
```

ii. Usuario: nombre, email.

```
10
11     /
12     public class Usuario {
13         // Atributos
14         private String nombre;
15         private String email;
16
17         // Constructor
18         public Usuario(String nombre, String email) {
19             this.nombre = nombre;
20             this.email = email;
21         }
22
23         // Getters y setters
24         public String getNombre() {
25             return nombre;
26         }
27         public void setNombre(String nombre) {
28             this.nombre = nombre;
29         }
30
31         public String getEmail() {
32             return email;
33         }
```

```
34
35     public void setEmail(String email) {
36         this.email = email;
37     }
38 }
39
```

iii. GeneradorQR->método: void generar(String valor, Usuario usuario)

```
8  /*
9   * @author agust
10  */
11 public class GeneradorQR {
12     // Método de creación dependencia de creación
13     // Recibimos el valor y el usuario y devolvemos un nuevo CódigoQR
14     public CódigoQR generar(String valor, Usuario usuario) {
15         // Creamos el código QR a partir del valor y el usuario
16         CódigoQR código = new CódigoQR(valor, usuario);
17         return código;
18     }
19 }
20 }
```

Main

```
14  /*
15   * @param args the command line arguments
16  */
17 public static void main(String[] args) {
18
19     // Creamos un usuario
20     Usuario usuario = new Usuario("Walter Frias", "walter.fr@utn.com");
21
22     // Creamos el generador de QR
23     GeneradorQR generador = new GeneradorQR();
24
25     // Generamos un código QR para ese usuario
26     CódigoQR código = generador.generar("ACCESO-YA", usuario);
27
28     // Mostramos el código QR generado
29     System.out.println(código);
30 }
```

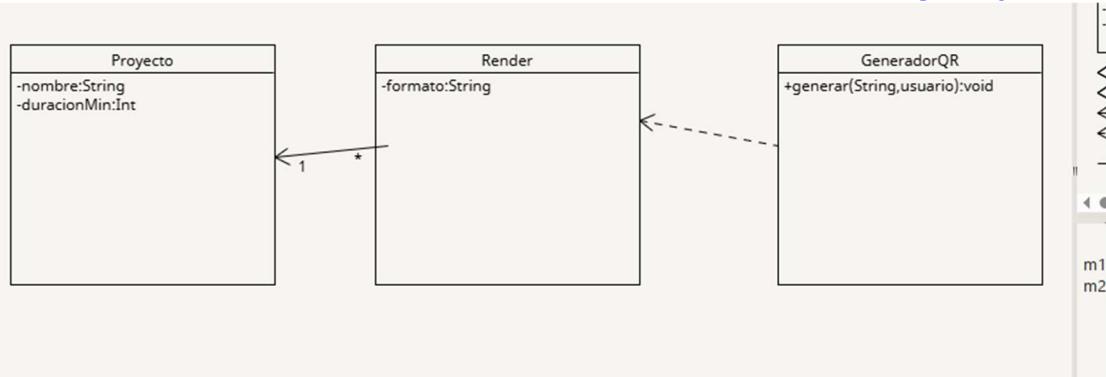
Print _Consola

```
Output - 13_GeneradorQR_Usuario_CódigoQR (run) ×
▶ run:
▶ CódigoQR{valor='ACCESO-YA', usuario=pkg13_generadorqr_usuario_códigoqr.Usuario@279f2327
BUILD SUCCESSFUL (total time: 0 seconds)
```

E

14. EditorVideo - Proyecto - Render

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



- Clases y atributos:
 - Render: formato.

```

9  * @author agust
10 */
11 public class Render {
12     // Atributos
13     private String formato; // ej: "MP4", "AVI"
14     private Proyecto proyecto; // asociación unidireccional a Proyecto
15
16     // Constructor
17     public Render(String formato, Proyecto proyecto) {
18         this.formato = formato;
19         this.proyecto = proyecto;
20     }
21
22     // Getters y setters
23     public String getFormato() {
24         return formato;
25     }
26
27     public void setFormato(String formato) {
28         this.formato = formato;
29     }
30
31     public Proyecto getProyecto() {
32         return proyecto;
33     }
34
35     public void setProyecto(Proyecto proyecto) {
36         this.proyecto = proyecto;
37     }
38
39     // Para mostrar el render
40     @Override
41     public String toString() {
42         return "Render{formato=" + formato + ", proyecto=" + proyecto + "}";
43     }
44 }
45
46
  
```

ii. Proyecto: nombre, duracionMin.

```
10  /public class Proyecto {  
11      //Atributos  
12      private String nombre;  
13      private int duracionMin; // duración en minutos  
14  
15      // Constructor  
16      public Proyecto(String nombre, int duracionMin) {  
17          this.nombre = nombre;  
18          this.duracionMin = duracionMin;  
19      }  
20  
21      // Getters y setters  
22      public String getNombre() {  
23          return nombre;  
24      }  
25  
26      public void setNombre(String nombre) {  
27          this.nombre = nombre;  
28      }  
29  }
```

```
30  public int getDuracionMin() {  
31      return duracionMin;  
32  }  
33  
34  public void setDuracionMin(int duracionMin) {  
35      this.duracionMin = duracionMin;  
36  }  
37  
38  // Para mostrar el proyecto  
39  @Override  
40  public String toString() {  
41      return "Proyecto{nombre=" + nombre + ", duracionMin=" + duracionMin + "}";  
42  }  
43  }  
44  }
```

iii. EditorVideo->método: void exportar(String formato, Proyecto proyecto)

```
9 * @author agust
10 */
11 public class EditorVideo {
12     // Método de creación (dependencia de creación)
13     // No devuelve nada (void), solo crea el Render y muestra un mensaje
14     public void exportar(String formato, Proyecto proyecto) {
15         // Creamos el render a partir del proyecto y el formato
16         Render render = new Render(formato, proyecto);
17
18         // Mostramos un mensaje de exportación
19         System.out.println("Exportando \\"+
20             + proyecto.getNombre() +
21             +" en el formato " + formato + "...");
22         System.out.println("Render generado: " + render);
23     }
24 }
25 }
```

Main

```
>Main.java [-/A] <--> EditorVideo.java [-/A] <--> Proyecto.java [-/A] <--> Render.java [-/A] <-->
```

Source History

```
10 */  
11 public class Main {  
12  
13     /**  
14      * @param args the command line arguments  
15     */  
16     public static void main(String[] args) {  
17         // Creamos un proyecto de video  
18         Proyecto proyecto = new Proyecto("Video Clip, 15 años de  
19  
20         // Creamos el editor de video  
21         EditorVideo editor = new EditorVideo();  
22  
23         // Exportamos el proyecto en un formato dado  
24         editor.exportar("MP4", proyecto);  
25     }  
26 }
```

221

```
Output - 14_EditorVideo_Proyecto_Render (run) x
run:
Exportando "Video Clip, 15 años de guadalupe" en el formato MP4...
Render generado: Render(formato='MP4', proyecto=Proyecto(nombre='Video Clip, 15 años de guadalupe', du
BUILD SUCCESSFUL (total time: 0 seconds)
```

CONCLUSIONES ESPERADAS

- Diferenciar claramente los tipos de relaciones entre clases (asociación, agregación, composición).
 - Representar las relaciones con la dirección adecuada en diagramas UML.
 - Comprender e implementar dependencias de uso y de creación.

- Aplicar relaciones 1 a 1 en el diseño e implementación de clases en Java.
- Reforzar el análisis de modelos orientados a objetos y la capacidad de abstracción.