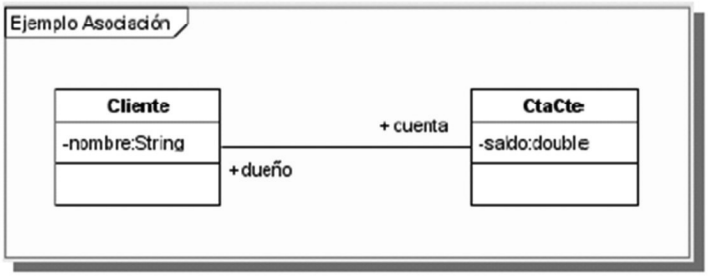
Asociación

- Bidireccional con multiplicidad 0..1 o 1



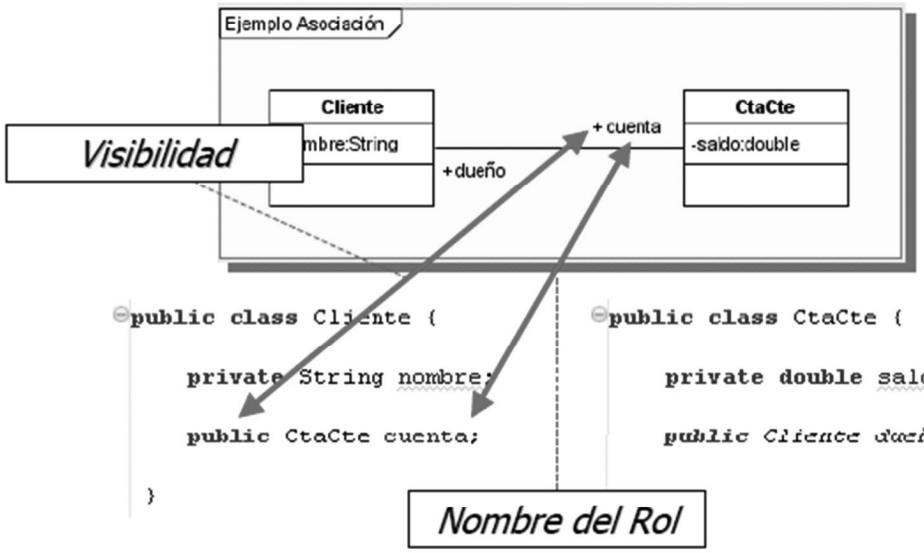

```
public class Cliente {

    private String nombre;

    public CtaCte cuenta;

}
```

```
public class CtaCte {

    private double saldo;

    public Cliente dueño;

}
```

La línea puede ser con flechas o sin ellas

Asociación

- Bidireccional con multiplicidad 0..1 o 1

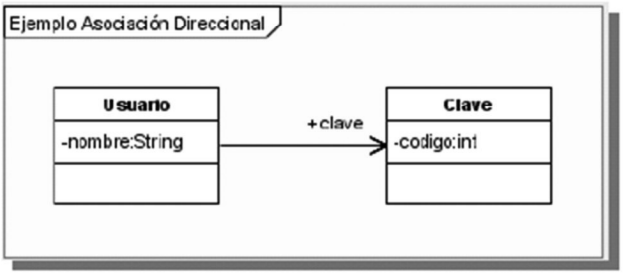



```
public class Cliente {

    private String nombre;

    public CtaCte cuenta;

}
```

```
public class CtaCte {

    private double saldo;

    public Cliente dueño;

}
```

Asociación

- Direccional con multiplicidad 0..1 o 1

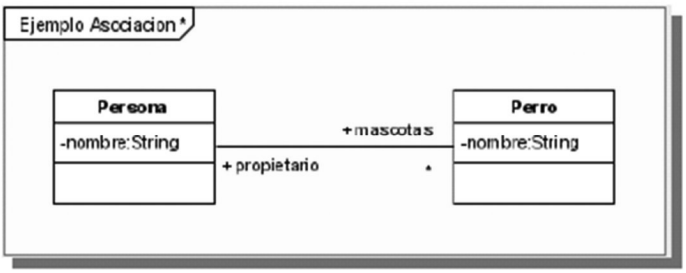



```
public class Usuario {

    private String nombre;

    public Clave clave;

}
```

```
public class Clave {

    private int codigo;

}
```

Asociación

- Bidireccional con multiplicidad *

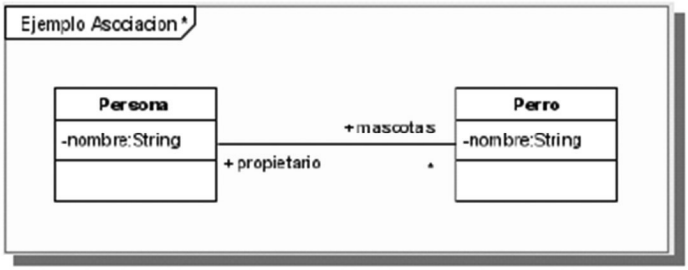



```
public class Persona {

    private String nombre;

    public java.util.Collection mascotas = new java.util.TreeSet();

}
```

```
public class Perro {

    private String nombre;

    public Persona propietario;

}
```

Asociación

- Bidireccional con multiplicidad *



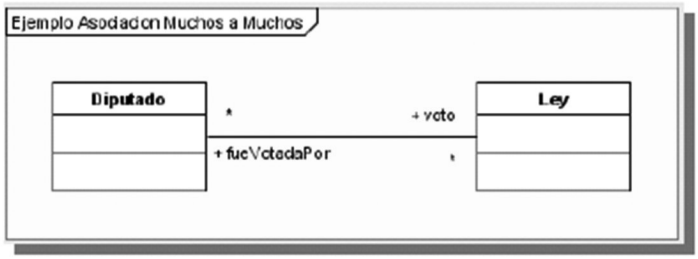

```
public class Persona {

    private String nombre;

    public java.util.Collection mascotas = new java.util.TreeSet();

}
```

```
public class Perro {

    private String nombre;

    public Persona propietario;

}
```

Decisión de Implementación

Asociación

- Bidireccional con multiplicidad *

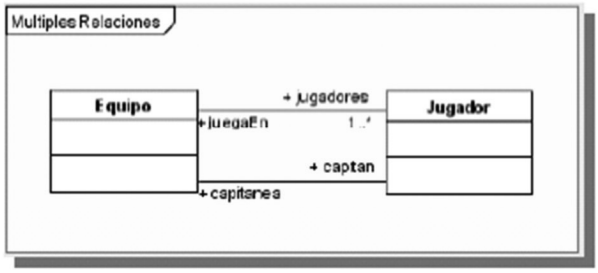



```
public class Diputado {

    public java.util.Collection voto = new java.util.TreeSet();

}
```

```
public class Ley {

    public java.util.Collection fueVotadaPor = new java.util.TreeSet();

}
```

Asociación

- ¿Con más de una relación?

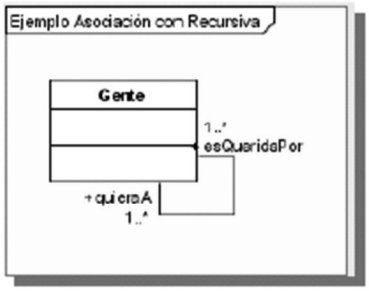



```
public class Equipo {

    public Jugador capitan;

    public java.util.Collection jugadores = new java.util.TreeSet();

}
```

```
public class Jugador {

    public Equipo capitanea;

    public Equipo juegaEn;

}
```

Asociación

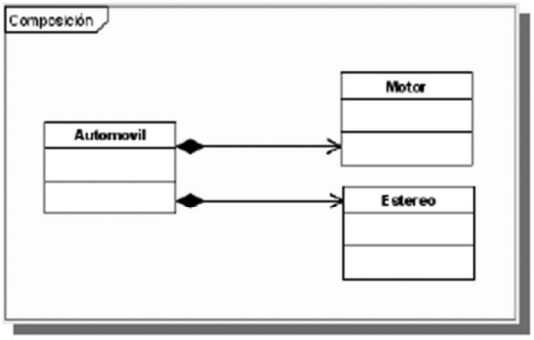
- ¿Y con esto?




```
public class Gente {

    public java.util.Collection quieraA = new java.util.TreeSet();

    public java.util.Collection esQueridaPor = new java.util.TreeSet();

}
```

Composición

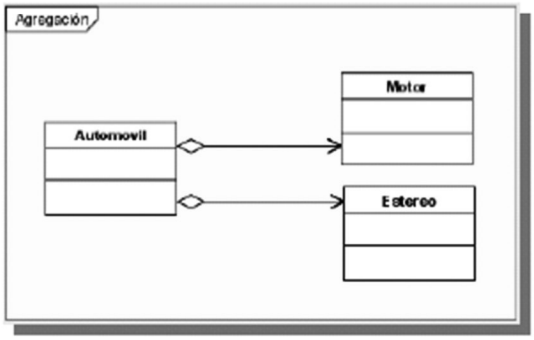
- Hay una dependencia en los ciclos de vida




```
public class Automovil {

    public Estereo estereo;
    public Motor motor;

    public Automovil() {
        estereo = new Estereo();
        motor = new Motor();
    }

}
```

Agregación

- La diferencia con la composición es en el modo en que se implementa




```
public class Automovil {

    public Estereo estereo;
    public Motor motor;

    public Automovil() {
    }

    public void ensamblar(Estereo e, Motor m) {
        estereo = e;
        motor = m;
    }

}
```