Asociación de objetos

La asociación implica que una clase esté "consciente" de otra y mantenga una referencia a ella, es decir, posea una propiedad del tipo de esa otra clase. Hasta ahora, hemos utilizado propiedades de clases predefinidas en Java, como String, Integer, Scanner, entre otras. Sin embargo, **la asociación implica el uso de propiedades que son instancias de clases personalizadas.**

Consideremos la relación entre una clase "Jugador" y una clase "Equipo". Un jugador pertenece a un equipo, pero incluso si el equipo se disuelve, los jugadores aún existirán.

Tomemos un ejemplo concreto:

```
public class Jugador {
    private String nombre;
    private Equipo equipo;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public Equipo getEquipo() {
        return equipo;
    }
    public void setEquipo(Equipo equipo) {
        this.equipo = equipo;
    }
}
```

```
public class Equipo {
    private String nombre;

    public String getNombre() {
```

```
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

💡Podríamos contemplar la relación en sentido inverso, teniendo un arreglo de jugadores dentro de la clase Equipo. Sin embargo, esto podría resultar en la necesidad de mantener un conjunto de jugadores sin equipo para preservar las instancias de jugadores que se eliminen de los equipos. Además, los arreglos tienen un tamaño fijo, lo que complicaría la gestión si quisiéramos agregar más jugadores que el tamaño original del array.

Agregación

La agregación es una forma de asociación en la que el ciclo de vida de un objeto no depende del otro. En Java, este tipo de relación ocurre de manera predeterminada, es decir, no es necesario añadir ninguna lógica adicional a nuestras clases, como Jugador y Equipo en el ejemplo anterior, para que la asociación sea de agregación.

Composición

La composición es otra forma de asociación en la que el ciclo de vida de una clase depende del otro. En este caso, los objetos contenidos no pueden existir independientemente del objeto contenedor. Si el objeto contenedor se destruye, los objetos contenidos también se destruyen.

En nuestro ejemplo, los jugadores pertenecen a un Equipo, lo que hace que el Equipo sea el objeto contenedor. Sin embargo, por razones de practicidad, hemos decidido establecer la relación con un atributo equipo dentro de la clase Jugador, en lugar de tener un atributo array de jugadores dentro de la clase Equipo.

Si quisiéramos que la asociación sea de composición, tendríamos que implementar una lógica en nuestro programa que garantice la eliminación de los objetos "contenidos" cuando se elimine el objeto "contenedor". Sin embargo, la estructura básica de las clases modelo permanece igual.

Más sobre relaciones

En las relaciones, ya sea composición o agregación, estas pueden ser de uno a uno, de cero a uno, de uno a muchos y de cero a muchos. El tipo de relación se refleja al utilizar el objeto como atributo en la clase receptora de la relación. Por ahora, nos concentraremos en trabajar sólo con relaciones uno a uno y uno a muchos, ya que son las más comunes.

UNO A UNO

Este tipo de relación se establece cuando hay una relación con un solo objeto. Por ejemplo, un curso tiene un único profesor.

```
public class Profesor {
    private String nombre;
    private String apellido;
}
```

```
public class Curso {
    private String nombre;
    private char division;
    private Profesor profesor;


    public Profesor getProfesor() {
        return profesor;
    }

    public void setNombre(Profesor p) {
        this.profesor = p;
    }
}
```

UNO A MUCHOS

En este caso, por cada objeto tenemos una relación con múltiples objetos de una clase. Por ejemplo, para un curso tenemos muchos estudiantes.

```
public class Estudiante {
    private String nombre;
    private String apellido;
}
```

```
public class Curso {
    private String nombre;
    private char division;
    private Estudiante[] estudiantes;
```

```
    public Estudiante[] getEstudiantes() {
        return estudiantes;
    }

    public void setEstudiantes(Estudiante[] e) {
        this.estudiantes = e;
    }
}
```